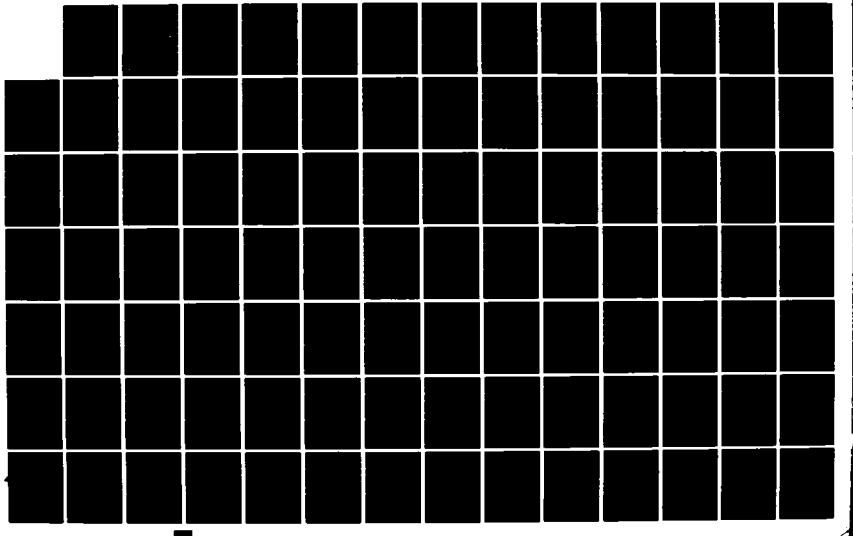
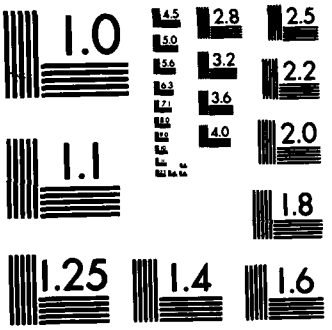
AD-A122 838 RESEARCH ON NARROWBAND COMMUNICATIONS(U) BOLT BERANEK 1/2
AND NEWMAN INC CAMBRIDGE MA S ROUCOS ET AL. NOV 82
BBN-5231 F19628-80-C-0165

UNCLASSIFIED F/G 17/2 NL





Bolt Beranek and Newman Inc.

Report No. 5231

Research on Narrowband Communications

Final Report

November 1982

Prepared for:
Defense Advanced Research Projects Agency

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 5231	2. GOVT ACCESSION NO. AD-A122838	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) RESEARCH ON NARROWBAND COMMUNICATIONS		5. TYPE OF REPORT & PERIOD COVERED Final Report 18 Aug 1980-30 Nov 1982 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Salim Roucos John Makhoul Richard Schwartz		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0165
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE November 1982 13. NUMBER OF PAGES 74
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Deputy for Electronic Technology (RADC/EEV) Hanscom AFB, MA 01731 Attn: Mr. Anton Segota		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Dept. of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Porjects Agency under ARPA Order No. 3515, AMD. 4.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech compression, linear prediciton, clustering, spectral template, vocoder, unsupervised learning, diphone, phonetic vocoder, phoneme recognition, time warping, segmentation, segment vocoder, segment quantization and space sampling.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We describe in this report the methods used to implement a very-low-rate (VLR) vocoder that transmits speech with a bit rate in the range of 100 to 200 b/s. The primary goal of this two-year project was for the vocoded speech to be intelligible in context, i.e., the vocoder can be used in a conversation.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

In the first year, we investigated two independent approaches for designing the VLR vocoder. The first approach was the phonetic vocoder in which the sequence of phonemes in the input speech must be automatically recognized. The phonetic vocoder uses a supervised training approach and requires a database of phonetically transcribed and hand labelled speech. The second approach uses vector quantization and Markov chain modeling to reduce the bit rate of an LPC vocoder from 2400 b/s to the range of 100-200 b/s. This latter approach is unsupervised and does not require any human effort in the training phase. The work on the above two approaches led to the formulation of the final system: the segment vocoder.

In the second year, the segment vocoder was implemented and tested. The segment vocoder models speech as a sequence of segments. A segment consists of a sequence of frames and has a duration comparable to the duration of a diphone. In the segment vocoder, a segment is determined automatically by a segmentation algorithm and does not require the labor intensive process of hand labelling. The work on vector quantization and Markov modeling determined that both the log-area-ratio (LAR) parameters representing a single frame of speech and the LARs of consecutive frames are statistically dependent. This statistical dependence has been exploited by quantizing a segment as a single unit in the segment vocoder.

The segment vocoder, operating in a single speaker mode, was demonstrated during the final ARPA NSC meeting in June, 1982. The vocoder used an average bit rate of 150 b/s to transmit the speech of a single speaker. The vocoded sentences were highly intelligible. The quality of the vocoded speech was quite close to the quality of an LPC synthesizer using unquantized parameters and therefore quite natural sounding.

During the first year, we also investigated the use of several techniques for multispeaker synthesis. The goal was to use a set of templates (segment templates or diphone templates) that were derived from one speaker to synthesize speech that sounded more like a new vocoder user. In this report we describe the above algorithms and present our results on the VLR vocoder.

Report No. 5231

RESEARCH IN NARROWBAND COMMUNICATIONS

Final Report

Authors: Salim Roucos, R.M. Schwartz, and John Makhoul

November 1982

Prepared for:
Defense Advanced Research Projects Agency

TABLE OF CONTENTS

LIST OF FIGURES

1. OVERVIEW

The primary goal of this two-year project was to demonstrate a very-low-rate (VLR) vocoder that transmits speech at a rate of 100 to 200 b/s. At these bit rates, the vocoded speech is required to be intelligible in context, i.e., the vocoder can be used in a conversation. The quality of the vocoded speech was to be as natural sounding as possible.

In the first year, we investigated two independent approaches for designing the VLR vocoder. The first approach was the phonetic vocoder in which the sequence of phonemes in the input speech must be automatically recognized. The phonetic vocoder uses a supervised training approach and requires a database of phonetically transcribed and hand labelled speech. The second approach uses vector quantization and Markov chain modeling to reduce the bit rate of an LPC vocoder from 2400 b/s to the range of 100-200 b/s. This latter approach is unsupervised and does not require any human effort in the training phase. The work on the above two approaches led to the formulation of the final system: the segment vocoder.

In the second year, the segment vocoder was implemented and tested. The segment vocoder models speech as a sequence of segments. A segment consists of a sequence of frames and has a

duration comparable to the duration of a diphone. In the segment vocoder, a segment is determined automatically by a segmentation algorithm and does not require the labor intensive process of hand labelling. The work on vector quantization and Markov modeling determined that both the log-area-ratio (LAR) parameters representing a single frame of speech and the LARs of consecutive frames are statistically dependent. This statistical dependence has been exploited by quantizing a segment as a single unit in the segment vocoder. We call this process segment quantization.

The segment vocoder, operating in a single speaker mode, was demonstrated during the final ARPA NSC meeting in June, 1982. The vocoder used an average bit rate of 150 b/s to transmit the speech of a single speaker. The vocoded sentences were highly intelligible. The quality of the vocoded speech was quite close to the quality of an LPC synthesizer using unquantized parameters and therefore quite natural sounding.

During the first year, we also investigated the use of several techniques for multispeaker synthesis. The goal was to use a set of templates (segment templates or diphone templates) that were derived from one speaker to synthesize speech that sounded more like a new vocoder user. By using an average vocal tract length normalization and a long term average spectrum normalization, the spectral parameters of the templates can be

modified to sound more like the new speaker. For those speakers whose speech was significantly different from the database talker, the resulting output speech sounded much more like the new intended speaker.

The final report is organized into five major Sections. In each Section we describe the work that was done on one major topic. These five topics are:

- Phonetic vocoder
- Vector quantization
- Markov chain models for speech
- Segment vocoder
- Multiple speaker synthesis

We summarize below the major issues and results of each Section. We have included in Appendix I of this report three conference papers that describe several aspects and results of the work performed under this project. The first two papers were presented at the International Conference on Acoustics, Speech and Signal Processing in Paris, 1982. The third paper was presented in Globecom-82, in Miami, 1982.

1.1 Phonetic vocoder

During the first year of the project, we performed several experiments with the phonetic vocoder approach to very-low-rate

vocoding. This approach uses an automatic speech recognition technique to transmit speech at 100 b/s. The speech recognizer uses a diphone network model of speech for the recognition process. A diphone is defined as the region from the middle of a phoneme to the middle of the following phoneme. Thus, we expect the diphone model to represent most of the coarticulatory effects of one phoneme on adjacent phonemes. Since not all diphones can follow a given diphone (two successive diphones must have a common phoneme), we use a diphone network to specify these sequential constraints.

The recognition process is a matching process. An input sentence is matched to the nearest path (using a spectral distance measure) in the diphone network. The sequence of diphones in the nearest path to the input is considered as the input diphone sequence. In the diphone network, we typically have several templates for each diphone. At the receiver, only one template per diphone is used in synthesis. Therefore the spectral error between the input and the synthesized output is quite large. But if the recognition process is highly accurate, the synthesized speech would be intelligible since the correct phoneme sequence in the input is reproduced in the synthesized output. We determined that a phoneme recognition rate of 80% is necessary to achieve the proper performance level in the

matching process. Due to computational limitations a beam search is used to determine the best matching path. A stack length of 600 simultaneous theories was found to be adequate. Increasing the stack length to 3000 did not improve the recognition rate significantly.

The major issues of the phonetic vocoder have been the amount of training data necessary to estimate the diphone model and how the training data is used. Obtaining a training data set requires a large human effort since we must segment and label continuous speech. A total of 5 minutes of speech was labelled. An initial estimate of the diphone network was based on one diphone template for each of 2800 diphones where each template is extracted from a carefully recorded nonsense syllable that contains the required diphone. This network is also used for synthesis. The phoneme recognition rate was 36% when the diphone network based on the nonsense syllables is used. By adding additional diphone templates extracted from continuous speech the performance improved to 62% when a total of 4200 templates were used. A significantly larger amount of training data is expected to improve the recognition performance. But, the human effort required to hand label the required database is prohibitively expensive. We therefore investigated an alternative approach to the phonetic vocoder that avoids the transcription and hand

labelling of speech. The work on the segment vocoder will be described in Section 5. In our work on the phonetic vocoder, we have evaluated several methods for using the additional diphone templates. These will be discussed in Section 2.

We also describe in Section 2, some variations on the phonetic vocoder that improved the intelligibility of the vocoder, with a moderate increase in the bit rate. In the phonetic vocoder, the synthesized diphone can have a rather different spectrum from the input diphone. The diphone template used for synthesis is not necessarily the nearest template to the input. To improve the spectral match between the input and the synthesized output, we specified which template was nearest to the input. This allophone vocoder requires an additional 30 b/s and has a slightly higher intelligibility than the phonetic vocoder. In order to improve the spectral match further, we did not use the network constraints, i.e., any diphone was allowed to follow a given diphone. This diphone vocoder has a bit rate around 200 b/s and is quite intelligible. The segment vocoder, described in Section 5, is an extension of the diphone vocoder that does not require any hand labelling of speech.

1.2 Vector Quantization

We describe in Section 3 several methods for quantizing the LAR parameters used to represent a single frame of speech. We compared several clustering algorithms for designing a vector quantizer. We found that a non-uniform binary clustering algorithm achieved a good performance with a large savings in the computational load as compared to the optimal K-means algorithm. We also used a model of optimal scalar quantization to evaluate the gain due to statistical dependence in vector quantization. In particular, we found that coding 14 LAR parameters of a single frame of speech required 10 bits for a vector quantizer instead of 15 bits for an optimal scalar quantizer for the same quantization error. Since the vector quantizer has a 30% lower bit rate than the optimal scalar quantizer, the former quantization scheme was used with a variable frame rate (VFR) algorithm to transmit the spectrum alone at 180 b/s (6 bits x 30 frames/s). This system yields intelligible speech for a single speaker and was used for the Markov chain modelling of speech. The work on vector quantization is described in detail in Section 4.

1.3 Markov Chain Models of Speech

To reduce the bit rate of an LPC vocoder that uses a 6 bit vector quantizer for the spectrum and a VFR algorithm with an

average frame rate of 30 frames/s, we used a Markov chain model of the sequence of quantized spectra. Since we expect that consecutive speech spectra to be statistically dependent, the Markov chain model, which uses the past to predict the future, can be used to reduce the bit rate required for coding the spectrum. A first-order chain reduced the entropy from 6 bits to 4.75 bits/transmission. Since this bit rate was still too high for the VLR vocoder, we needed to estimate a higher order Markov chain. To minimize the amount of data required to estimate high order models, we proposed two new Markov models. The variable resolution model was most effective and had an entropy of 4 bits/transmission when 256 states were used in the model. This work is described in Section 4.

1.4 Segment Vocoder

Vector quantization is an attractive method for quantizing a set of parameters when these parameters are statistically dependent (beyond correlation). We show in Section 3 that the LARs of a single frame of speech are statistically dependent. Also, the variable resolution Markov model, described in Section 4, demonstrated that consecutive spectra of speech are highly dependent. To exploit the above statistical dependencies, we use a vector quantizer for quantizing all parameters that represent

several consecutive frames of speech. These consecutive frames define a segment and the corresponding quantizer is called a segment quantizer. The segment vocoder which is described in Section 5, uses segment quantization to vocode speech at an average bit rate of 150 b/s. Our work in the phonetic vocoder and its variations guided our choice in defining a segment. We required the segment to have an average duration comparable to a phoneme's duration. We used a segmentation algorithm similar to phonetic segmentation algorithms. One of the most successful segmentation algorithms that we used, generated segments that are analogous to diphones. The corresponding segments were defined from the middle of a spectral steady state to the middle of the following steady state.

As we demonstrate in Section 5, the gain track and voicing pattern of a segment are highly dependent on the spectral sequence of the segment. If two segments are spectrally close then they generally have the same gain track and voicing pattern. Hence, these are not transmitted in the segment vocoder, the gain track and voicing pattern of the template is used at the receiver. Only a level adjustment of the gain track is transmitted for each segment.

We describe in Section 5, the segment vocoder and the techniques used for segmentation, segment quantization, and gain and pitch quantization.

1.5 Multispeaker Synthesis

In both the phonetic vocoder and the segment vocoder, the output speech sounds like the speaker used to generate the templates. To make the output speech sound more like a new vocoder user, we investigated several methods for transforming the template data base. The transformation was to be determined using a small amount of training data from the new speaker.

The basic procedure was applied on the phonetic synthesis part of the phonetic vocoder. We required the speaker to speak for a period from 20 to 60 seconds. The speech from the new speaker was analyzed to extract several parameters which were used to transform the diphone templates to make the phonetic synthesizer sound more like the new speaker. The parameters used for the transformation are the average vocal tract length of the new speaker and the long term average spectrum for voiced, unvoiced and silence portions of the new speaker's speech. The use of these parameters is described in Section 6. We have found the speaker transformation to be effective particularly when the new speaker sounded quite differently from the database talker.

2. PHONETIC VOCODER

During the first year of this contract, we performed several experiments with the phonetic vocoder approach to very-low-rate vocoding. In this Section we will first review briefly the basic operation of the phonetic vocoder. Then, we will describe those experiments performed in an effort to make the phonetic recognition performance high enough such that the resynthesized speech was intelligible.

2.1 Methods Used

Figure 2.1 shows a block diagram of the phonetic vocoder. This figure shows that the input speech is analyzed to produce a set of phonemes, phoneme durations, and pitch values. A phoneme and its associated value of duration and pitch is called a "triplet". Speech rates are typically about 12 phonemes per second, and since each triplet can be encoded into 8 bits, the data rate in the transmission channel is about 100 bits per second. Once the triplets are decoded at the receiving end, a phonetic synthesizer reconstructs the original speech.

The basic model of speech that we chose to use in the phonetic vocoder is the diphone model. A diphone is defined as

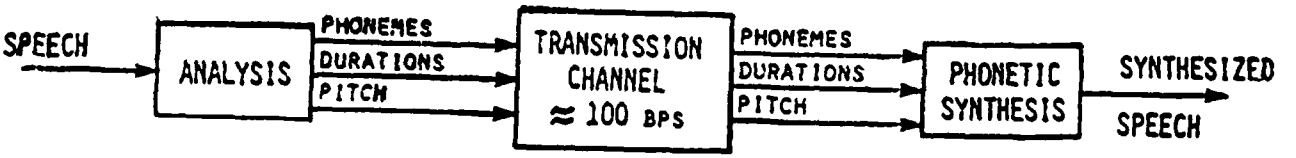


Fig. 1. Very-low-rate (VLR) phonetic vocoder.

the region from the middle of one phoneme to the middle of the next phoneme. Thus, the diphone model directly represents much of the coarticulatory effect of one phoneme on the adjacent phonemes. Both the analysis and synthesis components of the phonetic vocoder require a large database of diphone templates.

The phonetic synthesis program translates a sequence of phonemes into the corresponding diphone sequence, and then constructs LPC parameter tracks by concatenating the diphone templates for those diphones. The program also uses appropriate time-warping, and smoothing algorithms that are designed to maximize the naturalness of the output speech.

The phonetic recognizer uses the same diphone model to recognize the sequence of phonemes. The diphone templates are compiled into a network that constrains the sequence of diphones. That is, diphone A-B can only be followed by a diphone that starts with phoneme B. The diphone network consists of nodes and directed arcs. An example of a simple network is shown in Figure 2.2. There are two types of nodes: phone nodes and spectrum nodes. The phone nodes (shown as labelled circles) correspond to the midpoints of the phones; there is one such node for each phone. These phone nodes are connected by diphone templates. Each diphone template is represented in the network as a sequence of spectrum nodes (shown as dots). When two or more consecutive

spectra in the original diphone template are very similar, they are represented by a single spectrum node in the network. The open dots indicate the first spectrum node in the original diphone template that is at or past the labelled phone boundary. Note that, in Fig. 2.2, the diphone template P1-P2 is distinct from the template P2-P1. Also note the possibility of diphones of the type P1-P1. The network allows for multiple templates going from one phone to another (e.g., P2-P1). Branching and merging of paths within a template is also allowed (e.g., P1-P3). The network also allows the specification of diphones in context. The phone node P4/&P3 represents the phone P4 followed only by P3. Thus the template P2-P4/&P3 is different from the unconditioned template P2-P4. Finally, the network allows for sequences of diphones, for example in clusters, to be treated as an independent unit altogether (P1-P5*-P3). The generation and training of the network is discussed below.

Each spectrum node in a diphone template consists of a model for both the spectrum and the duration. The spectral model is represented by means and standard deviations for all 14 log-area-ratio (LAR) coefficients and gain. The duration of a node is defined as the number of frames of input aligned with the node. Each node contains a smoothed probability density of the duration of the node based on actual alignments during training. Each

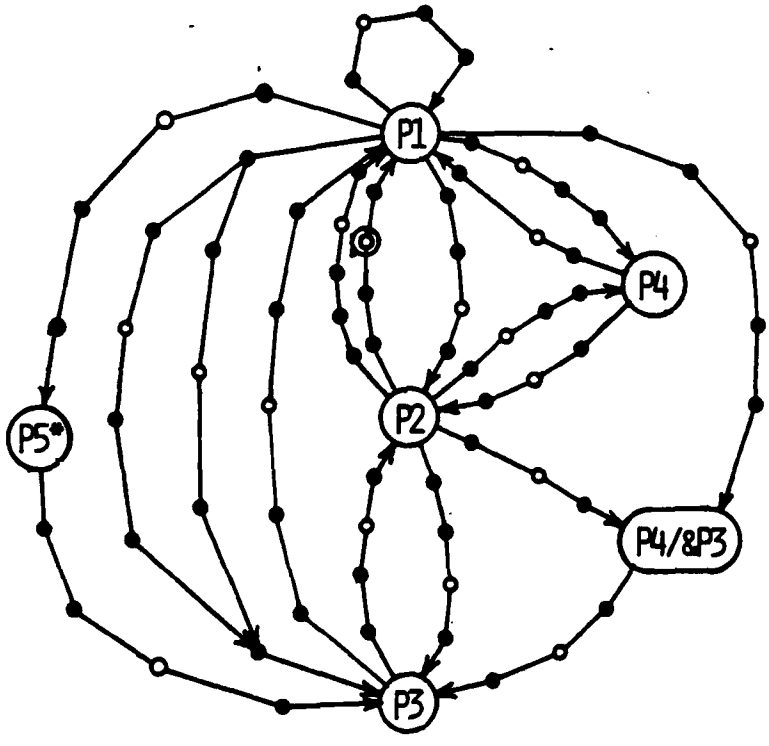


Fig. 2. Example of diphone template network for five phonemes.

spectrum node has an implied self-loop, so that the diphone matcher (which will be discussed below) can align several input frames to one spectral node.

The network matcher uses a stack-based dynamic programming algorithm which attempts to find the sequence of templates in the network that best matches the input according to a scoring algorithm. This score includes components due to the spectrum (LAR's), the durations, and also the probability of the associated phoneme sequence. In our feasibility study for this project, we found that by inclusion of first order phoneme statistics (probability of phoneme pairs or diphones) into the recognition process, phoneme identification accuracy improved by 15%. The main effect of the inclusion of phoneme pair probabilities in this program was that it greatly reduced the number of extraneous phonemes inserted into the output, but did not substantially change the probability of the correct phoneme appearing in the output.

The basic operation of the program begins by updating each "theory" by the addition of the newest input frame. A theory consists of a detailed account of how a sequence of input frames is aligned with the network, along with a total score for that correspondence. Each old theory will generate several new theories. First, a theory in which the new input frame is

matched against the same network node as the previous input frame. Second, a theory for each possible following node in the network. And third, for each pair of two following nodes. After all old theories have been expanded into new ones, the program keeps all theories that are within a score threshold of the best theory ("beam search"), and also limits the number of theories to a maximum number ("bounded breadth search"). All theories are kept in a tree, such that it is possible to determine, at any time, whether all theories have a common beginning. When they do, that part of the theories that agree can be output. Thus, there is a short lag (an average of 30 frames) between the input and the output of the chosen answer. We have found that preserving several hundred theories in the stack seems to result in an answer that has a score close to the score obtained with a much larger stack. Therefore, we conclude that the pruning is not often eliminating theories that would eventually score better than theories that are kept.

2.2 Training the Network

Much of the work on the phonetic vocoder was devoted to developing different algorithms for training the network model for speech. The first method of updating the network that we implemented relies on augmenting the network with additional

diphone branches. In this procedure, we use the transcription of the training data, together with the network compiler program, to create new alternate diphone templates. Each of the templates is independent, except that all the templates for a single diphone start and end at the same phoneme nodes.

The second method is more automated. The automatic training capability of the matcher allows the researcher to input to the matcher a sentence that has been phonetically transcribed. The input transcription includes both phonetic labels and may include the time of each phoneme. The phoneme may be left unspecified where desired, and the times may be specified as ranges if the best boundary location is not clear. The matcher then finds the best alignment (and corresponding score) of the input utterance against the network under the constraint of the transcription. Once completed, the matcher uses the input utterance to "train" the network. Those portions of the input utterance that are similar (closer than a threshold) to the path in the network that it was aligned with are used during the training procedure by updating the statistics of that closest path in the network to include the input utterance parameters. The statistics of the network path that are modified include the the means and variances of the LARs and the PDFs of the frame durations. The remaining portions of the input utterance, those that are not

very similar to the aligned path in the network, are used to add alternate branches to the network. The parameters of those portions of the input utterance are used to create new branches of the network. By this procedure of updating network statistics and augmenting the network with new branches, we ensure that the network can match any speech from the training data within the specified error threshold. However, the amount of training data required such that the network will have sufficient paths and accurate statistics to model arbitrary input utterances well may be excessive.

There are three differences between the augmentation algorithm and the automatic training method:

1. In the augment mode, the entire diphone is always added as an alternate path.

2. In augment mode, the compiler assumes that the diphone boundary is at the middle of the labelled phone (which is a good heuristic) rather than letting the program assign the diphone boundary where it chooses.

3. In augment mode, there are no a priori probabilities assigned to paths. This differs from the automatic training mode where several paths may be "averaged" together. These a priori probabilities, however, are not currently used by the matcher.

To evaluate the above training methods, we "trained" the network on several sentences using each training method, and then tested the updated (trained) network using several other

sentences. A comparison of the results using the training algorithm and augmenting algorithm showed 8% better phoneme recognition with the augmentation method. As a result of this experiment, we chose to use our available training data with the augment algorithm.

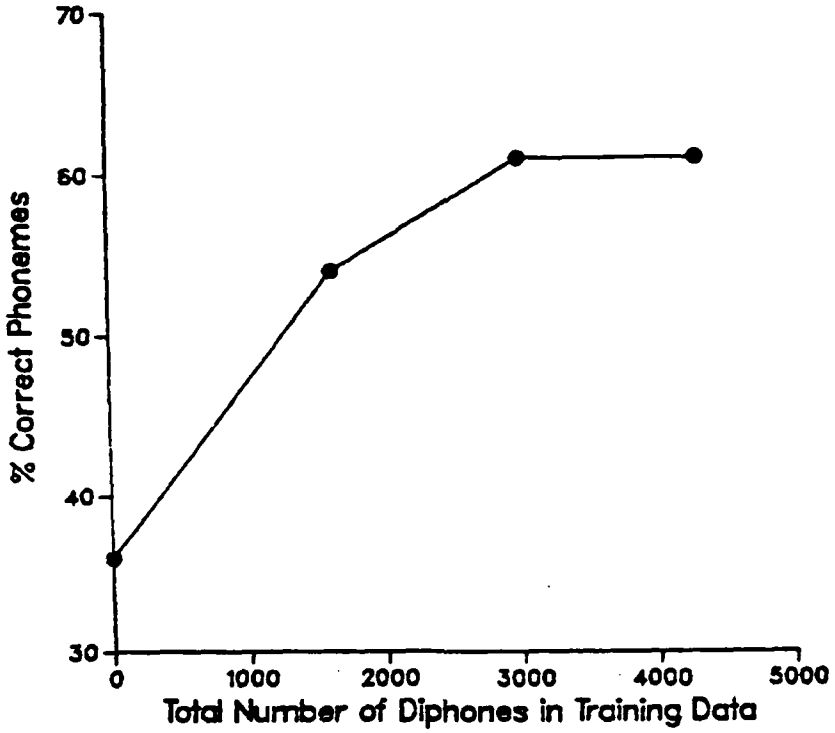
2.3 Recognition Improvement with Training

As mentioned above, it is necessary to train the network on natural speech, so that it contains a model for any of the many ways the different diphones can be pronounced. We recorded, digitized, and carefully transcribed 255 sentences of varying lengths. This produced about 4200 phonemes of training data. We then divided the training speech into three sets of approximately 1400 phonemes each. These were used incrementally to produce three diphone networks with different numbers of alternate paths. Thus, there were four different diphone networks. The first network had just one sample of each diphone taken from the phonetic synthesis database of nonsense utterances. We shall call this network "untrained." For each of the other three diphone networks, we determined the total number of diphones used to train it, the number of unique diphones used to train it (i.e., the number of diphones for which there was now at least one additional template), and the percentage of correctly recognized phonemes.

The test material consisted of 10 new sentences from the Harvard phonetically balanced list. These sentences had not been used in training. The total number of phonemes in the test sentences was 234.

Figure 3 shows the recognition performance as a function of the amount of training. Performance is given as a function of each of the two parameters described above: the total number of training diphones and the number of distinct training diphones. As the figure shows, the recognition performance improves considerably with additional training, improving from a recognition accuracy of 36% correct with no training (the "untrained" network) to 61% correct with 3000 total diphones of training). However, as the last point indicates, further training by the network augmentation method does not seem to make any significant improvement.

Careful examination of the training data indicated that even though only approximately 1200 of the 2800 possible diphones in the network had been augmented by the training with one or more alternate paths, over 90% of those diphones appearing in the test sentences were of diphones that had been augmented by additional paths. Thus, adding additional paths to diphones that were not needed in the test would not help at all. We looked at the subset of phonemes in the test for which two conditions were met:



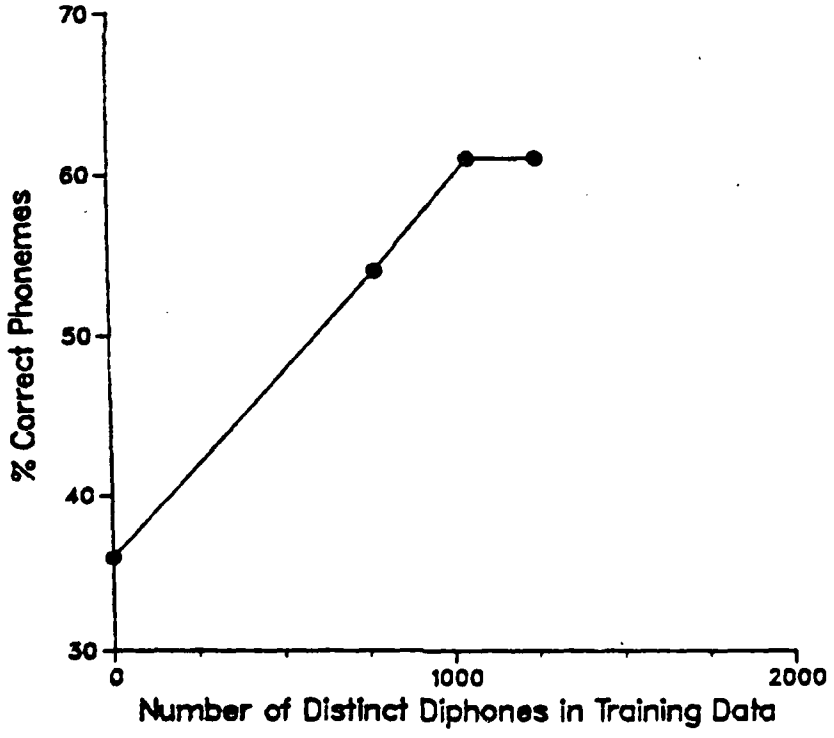


Fig. 3. The effect of training on phoneme recognition accuracy.

(1) the matcher had correctly identified both adjacent phonemes, and (2) the two diphones that span the phoneme had been trained. That is, if the correct phoneme string in the test sentence were

A B C

we only considered phoneme B if both A and C were correctly recognized, and the diphones A-B and B-C had been augmented by training. In these cases, we found that 85% of the phonemes were correctly recognized. This result indicates that the matcher tends to get long strings of phonemes correct. When a phoneme is incorrectly identified, it will usually be part of a string of several contiguous, incorrectly identified phonemes. It also suggests that if there were much more training, the performance might improve considerably. Unfortunately, this may be an inherent quality of a matcher such as ours that finds a globally optimal scoring path.

2.4 Conclusion

The primary conclusion from this project is that this method of VLR vocoding has the possibility of achieving very low data rates, but will need very large amounts of manually transcribed data before the phoneme recognition rate is high enough to make the output speech intelligible. Another problem with the use of the phonetic recognition and resynthesis for a vocoder was that

if we had multiple templates in the recognition network, but only one template for each diphone in the phonetic synthesis program, the output speech was no longer guaranteed to be spectrally close to the input speech. This realization prompted two experiments, which eventually led to the design and implementation of the Segment Vocoder, which will be discussed in a later Section.

2.4.1 Allophone Vocoder

To increase the intelligibility of the phonetic vocoder we considered transmitting extra information with each phone, specifying the identity of the actual diphone template that matched best by specifying in each case. Assuming 12 phones/second, and 8 templates/diphone, this would require only an additional 12x3=36 b/s. We call this vocoder an "Allophone Vocoder." An allophone is one of many possible variations in the way of pronouncing a phone. Although the diphone template network is identical to that used for the phonetic vocoder, the allophone vocoder does not use the many-to-one mapping discussed above. The vocoder synthesizes the spectral sequence - consistent with the network constraints - that is closest to the input spectral sequence, according to the distance metric used.

We found the output speech from the allophone vocoder to be substantially more intelligible than that from the phonetic

vocoder. However, due to the constraints of the network, the "nearest" spectral sequence chosen was often quite far from the input sequence, resulting in some intelligibility problems.

To further improve the intelligibility of the vocoded speech we needed to decrease the error between the input spectra and the synthesized spectra. The network constrains the sequence of diphones in such a way that taken together, the diphones form a phone sequence. A diphone template ending with a particular phone can be followed only by one of the diphone templates that begins with that same phone.

2.4.2 Diphone Vocoder

To decrease the spectral match error (still using the same set of diphone templates) we relaxed the constraint on the sequence of diphone templates that was imposed by the network. Thus, any diphone template could be followed by any other diphone template. This doubled the number of bits needed to transmit the sequence of diphone templates, bringing the total transmission rate up to about 200 b/s. (The source information still requires approximately the same number of bits.)

The result was that the spectral error decreased by 20% and the intelligibility improved to the point where most listeners

understood practically all the words and felt that this diphone vocoder could result in a usable speech transmission system.

Although the sequence of diphone templates transmitted by the diphone vocoder does not necessarily correspond closely to the "ideal" phonetic sequence, the spectra being synthesized are close enough to the input spectra so that (as with a conventional LPC vocoder) the human listener can make sense out of the speech. In other words, unless the required transmission rate is so low that only recognition methods are practical (below 130 b/s), it is more efficient, at this time, for the vocoder to simply do the best possible job of synthesizing a spectral sequence that sounds like the input sequence and leave the phone recognition to the human listener.

The diphone vocoder still has one significant drawback: the large amount of human effort required to transcribe a large data base of diphone templates. In the following Section we discuss a method that avoids this problem while maintaining all the advantages of the diphone vocoder.

3. VECTOR QUANTIZATION

3.1 Introduction

We describe in this chapter several methods for quantizing the log-area-ratio parameters (LARs) used to represent a single frame of speech. These methods were investigated in order to determine which methods will be most effective for reducing the bit rate of an LPC vocoder from 2400 b/s to the range from 100 to 200 b/s.

We compared several clustering algorithms for vector quantization. We found that a non-uniform binary clustering algorithm yields an acceptable performance with a significant reduction in the computational load over the optimal K-means algorithm. We also compared vector quantization to optimal scalar quantization of the LARs. We found that a scalar quantizer required 15 bits for quantizing 14 LARs whereas the vector quantizer required 10 bits for the same quantization error, a savings of 30% in bit rate. These results and several others will be discussed in more detail in the following Sections.

3.2 Optimal scalar Quantization

In the Government's LPC-10 standard vocoder, each LAR parameter is quantized separately using a uniform quantizer. In this Section, we describe the optimal scalar quantizer for n jointly gaussian parameters. The performance of the optimal scalar quantizer will be compared to that of a vector quantizer for quantizing the LARs representing speech in Section 3.4.

The optimal scalar quantizer for a set of n parameters, represented by a vector $\underline{x}$, minimizes the total mean square quantization error of all parameters for a given number of bits b. The n parameters are assumed to be jointly Gaussian. The optimal scalar quantizer consists of the following three steps:

i) Parameter decorrelation
ii) Bit allocation
iii) Scalar quantization

We describe each of these steps below:

Parameter Decorrelation: Let Q be the matrix whose columns are the eigenvectors of the covariance matrix C of the Gaussian vector $\underline{x}$. The new parameter vector $\underline{y} = Q'\underline{x}$ will have uncorrelated components, where Q' is the matrix transpose of

Q. The transformation by Q' corresponds to a pure rotation of the vector $\underline{x}$.

Bit Allocation: The second step is to allocate the given b bits to the components of the uncorrelated vector $\underline{y}$. In [1] we showed that the optimal bit allocation is such that each component gets the number of bits necessary for the resulting quantization error to be equal for all components whenever possible. In that case, the savings due to bit allocation is:

$$\Delta = \frac{n}{2} \log_2 (\frac{a}{g}) \qquad 4.1$$

where a is the arithmetic mean of the variances of all the components of $\underline{y}$ and g is their geometric mean.

Scalar Quantization: The third and final step is to perform the scalar quantization of each of the components $\{y_i\}_{i=1}^{n}$ using the corresponding allocated b_i bits. Here one simply uses a Max quantizer [2] designed for each component.

In the application of optimal scalar quantization to the LARs, one estimates the covariance matrix C from a training set of observed LAR vectors of speech. Then using the eigenvector matrix Q, the LAR vector $\underline{x}$ is rotated to obtain the uncorrelated vector $\underline{y}$.

Given b bits one determines the bit allocation and the n Max quantizers by the following process consisting of b steps:

> Initially all n components have zero bits allocated to them and the quantization error is equal to their variances. For each bit from 1 to b, we do the following operations.
>
> We allocate an additional bit to each component and redesign the n Max quantizers with the new bit allocation (i.e., using one more bit). Then, we determine the component that has to largest decrease in quantization error due to this additional bit. The additional bit is therefore allocated to this component.
>
> The above process is repeated b times until all b bits are allocated. At the end of this process we will have n Max quantizers each using b_i bits such that a total b bits are used in quantizing the n LARs.

The above process is optimal for jointly Gaussian random variables. In that case, one can show that the n Max quantizers differ only by a scaling factor. Since the LARs of speech are not jointly Gaussian, the n Max quantizers will differ by more than a scaling factor. In this case, we expect that the resulting scalar quantizer to be near optimal. We have found that the eigenvector rotation saves 3 bits in quantizing the 14 LARs. For a typical LPC vocoder operating at 2400 b/s, nearly 40 bits are used in quantizing the LARs. In this case the savings due to the

rotation is not very important. But for the very-low-rate vocoder, we expect to use 10 to 15 bits for the LAR vector so that the savings of 3 bit due to the eigenvector rotation is necessary. We will compare optimal scalar quantization to vector quantization in Section 3.4.

3.3 Clustering of Speech Spectra

Since we expect the LAR parameters of speech to exhibit a statistical dependence that is beyond correlation, we evaluated the use of vector quantization for quantizing the LAR vector. The vector quantizers that we evaluated were all based on the application of a clustering algorithm on a training data set of observed LAR vectors of speech.

An M-level, n-dimensional vector quantizer is defined by a partition $P=\{C_i; i=1,M\}$ of the space of all possible input vectors into M disjoint regions, each denoted by C_i. A template vector $\underline{z}_i$ is also defined for each region C_i. The input vector $\underline{x}$ is quantized into the template $\underline{z}_i$ if the vector $\underline{x}$ belongs to the region C_i.

A non-negative distortion measure, denoted by $d(\underline{x},\underline{z})$, is used as an objective measure of the loss in accuracy in representing an input vector $\underline{x}$ by a template $\underline{z}$. An optimal

vector quantizer must satisfy the following two necessary conditions:

Condition 1: Minimum distance classification.

The shape of the regions C_i must guarantee that an input vector is quantized to the nearest template.

$$\underline{x} \in C_i \Leftrightarrow d(\underline{x},\underline{z}_i) \leq d(\underline{x},\underline{z}_j) \quad \text{for } 1 \leq j \leq M \tag{4.2}$$

For the Euclidean distance measure the regions are bounded by hyperplanes.

Condition 2:Template Selection

The templates of an optimal vector quantizer must minimize the average distortion of their corresponding regions, i.e., the template $\underline{z}_i$ of the region C_i must minimize

$$\underset{z}{\text{minimize}} \int_{C_i} d(\underline{x},\underline{z})\, p(\underline{x})\, d\underline{x} \tag{4.3}$$

where $p(\underline{x})$ is the probability density function $\underline{x}$ and the integral is over the region C_i.

The above two conditions are necessary but not sufficient

for an optimal vector quantizer. These two conditions have been used to define an iterative clustering algorithm, called the K-means algorithm, that has been used to design vector quantizers for the LPC models of speech.

3.4 K-Means Algorithm

The K-means algorithm has been extensively used in pattern recognition as a clustering algorithm. Using a training set of observed LAR vectors, the K-means algorithm is a hill climbing algorithm that determines a set of K clusters (in our case K=M) that minimizes the clustering criterion. Each cluster will be represented by a single template. We use the average mean square quantization error as a clustering criterion.

The algorithm is described in detail in [3]. We present below a brief description of the K-means algorithm when the Euclidean distance on LARs is used:

1. Choose by some adequate method an initial set of M templates.
2. Classification: Classify all vectors in the training data set to the nearest template. A set of M clusters in thereby obtained where each cluster consists of all vectors classified to a given template.
3. Template updating: For each cluster a new template is obtained by averaging all vectors in the cluster.

4. Repeat steps 2 and 3 until the algorithm converges.

The algorithm is guaranteed to converge to a local minimum of the mean square error. We use a binary clustering algorithm, described in the following Section to determine a set of initial templates. In this case, the K-means algorithm converges in few iterations and usually 5 iterations are sufficient.

The major disadvantage of the K-means algorithm is the large computational load required. To quantize an input vector, M distance calculations are needed where $M=2^b$ and b is the number of bits used to transmit the LPC spectrum. Typically we use 10 bits for the spectrum for an LPC vocoder that operates between 200 and 400 b/s. Hence, 1000 distance calculations are needed for each input spectrum. In the next Section, we describe a binary clustering algorithm that only requires 20 distance calculations with a minimal increase in the quantization error. The computational load is reduced by a factor of 50. The binary clustering algorithm requires 2^b distance calculations instead of 2b required for the K-means algorithm.

3.5 Binary Clustering

To avoid the computational load the K-means algorithm, we used a hierarchical clustering algorithm. We present two binary clustering algorithms.

3.5.1 Uniform binary clustering

The binary clustering is applied sequentially in the following manner on a training data set. Initially, the training data set is divided into two clusters using the K-means algorithm (where K=2). Then, each cluster is further subdivided into two clusters. This process can be represented by a uniform binary tree where the root node corresponds to all the training data. The two sons of the root node correspond to the first two clusters. Then at each level, each node will have two sons corresponding to the clusters obtained by subdividing the cluster of the parent node. The process of subdivision is continued until the desired number of clusters is obtained. The K-means with K=2 is used for every subdivision.

To complete the specification of the binary clustering algorithm, we need to describe how the initial set of two templates is obtained when the K-means algorithm is used to subdivide a given cluster into two clusters. We used the following procedure. The mean vector of the parent cluster and the LAR component with largest variance are determined. Then, a hyperplane perpendicular to the component with largest variance and going thru the mean is used to divide the cluster into two. The means of the resulting two clusters are used as the initial set of two templates for the K-means (K=2) algorithm. For Gaussian clusters, this initial division is optimal.

4.5.2 Non-uniform binary clustering

For each additional bit, the uniform binary clustering algorithm divides all the clusters at a given depth on the binary tree into two clusters. This uniform binary subdivision divides all clusters whatever their contribution to the quantization error, small or large. A more effective algorithm is to adaptively divide the clusters that have the largest contribution to the quantization error while not subdividing those that have the smallest contribution.

The non-uniform binary clustering algorithm divides sequentially the cluster that has the largest contribution to the mean square error.This sequential process is performed until the required number of clusters is obtained. In general, the resulting binary tree is non-uniform, i.e., some clusters have more subdivisions than others. We expect this algorithm to have a smaller quantization error than the uniform binary for the same bit rate. We compared the performance of these two algorithms on a database of 14-dimensional LAR vectors of speech.

Using the mean square error on LARs, we compared the uniform binary clustering, non-uniform binary clustering and the K-means clustering algorithms. We have found that the non-uniform tree saves 0.5 bits over the uniform tree for the same mean square

error. The mean-square error of the non-uniform binary clustering and the K-means algorithm for a single speaker database is shown in Fig. 3.1. The non-uniform tree requires only 0.5 bits more than the K-means algorithm for the same quantization error. The small increase in bit rate of the non-uniform binary clustering as compared to the K-means algorithm is acceptable given the large savings in computation by factor of 50 when a 10- bit codebook is used. We now routinely use the non-uniform binary clustering for designing our VLR LPC vocoders. In figure 3.1, we also show the mean-square error of the non-uniform binary clustering on an all male multispeaker database. We find that an additional 0.7 bits are needed for the multispeaker quantizer to have the same single speaker quantization error. For the non-uniform binary clustering, we have used several criteria for selecting which cluster to subdivide next. We describe our results on this topic in the next Section.

3.5.2 Cluster Splitting Selection

The criterion used for selecting which cluster to subdivide next in the non-uniform binary clustering can be varied as described in the [3]. We found that choosing the cluster which has the largest mean square error for further subdivision yields the best vocoded speech quality. We note that this vocoder has a

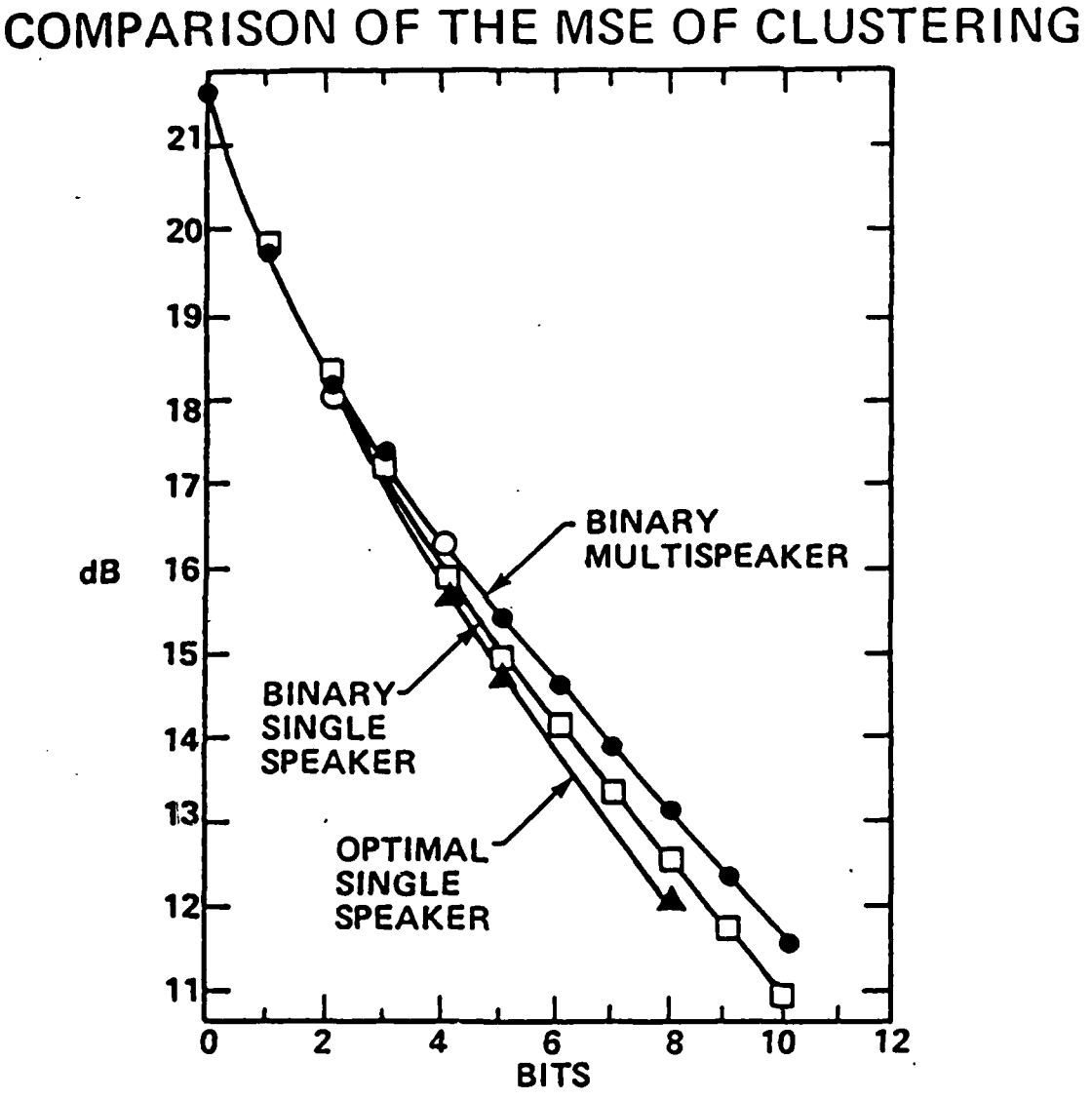


Fig. 4. Mean-square quantization error for non-uniform clustering, K-means clustering for a single speaker data.

slightly higher quantization error (+19%) than the optimal non-uniform binary clustering described in the previous paragraph. The optimal non-uniform binary tree chooses the cluster with the largest total swared error given by $N\bar{e}^2$ for further subdivision. The mean square error of the cluster is given by $\bar{e}^2$ and N is the number of vectors in the cluster. However, the perceptual difference in quality is rather small.

3.5.3 Distance Measures

We have compared the Euclidean distance on LARs with the Itakura-Saito distortion measure as described in [3] using the K-means algorithm. We compared two vocoders that used 8 bits for coding the LPC Spectrum of speech without pre-emphasis. Both vocoders used unquantized voicing, pitch and gain. The speech quality and intelligibility seemed to be similar for both vocoders in an informal listening test. Recently a more comprehensive study showed that the two distortion measures are quite similar in performance. We also found that using preemphasis with the Euclidean distance measure reduced the speech quality when compared with the Euclidean distance on LARs of non-preemphasized speech. The degradation can be characterized as an increase in roughness.

Finally, for the above distance measures we found that the

template vector of a cluster is obtained by an averaging process in the right domain as presented in [1]. For the Euclidean distance, the template is the average of all LAR vectors in a cluster. For the Itakura-Saito distance a weighted average of the autocorrelation matrices of all LPC spectra in a cluster is used to determine the template. The weight is the inverse of the prediction gain V_p as discussed in [3].

3.6 Comparison of Scalar and Vector Quantization

The major justification for using vector quantization instead of scalar quantization for speech compression has been based on the expected superior performance of the former method due to the statistical dependence of the spectral parameters of a frame of speech. We have seen that parameter correlation does not contribute to a difference in performance between vector and optimal scalar quantization. Hence, we have to determine if speech exhibits any statistical dependence other than correlation in order to justify the use of vector quantization. To estimate the savings in bit rate due to statistical dependence, we compared a vector quantizer with an optimal scalar quantizer for a data base of speech spectra represented by 14 LARs. The Euclidean distance was used to measure the quantization error. The mean square quantization error for both quantizers is shown

in Fig. 3.2. The top horizontal axis shows the cumulative bit allcoation for the eigenvector that is receuving the additional bit. We found that the vector quantizer was better than the scalar quantizer. The mean-square error of the 10-bit vector quantizer was equal to that of the 15-bit optimal scalar, a savings of 5 bits.

The advantage of vector quantization over optimal scalar quantization (a gain of 5 bits for the same mean-square error) is most significant for very-low-rate vocoding of speech. The size of available data sets limits of the bit rate of optimal vector quantizers to about 10 to 12 bits. For higher bit rates, suboptimal vector quantizers such as cascaded clustering which is described below may be used. However, the resulting loss in optimality reduces significantly the advantage of vector quantization over optimal scalar quantization. When we compared the two methods (cascaded and scalar) at 30 bits, we found cascaded vector quantization to be less robust than optimal scalar which resulted in the same performance for both methods. Therefore, at these higher bit rates, a scalar quantization methods would be most effective.

Recent published results [6] using the Itakura-Saito distance claim an advantage of 14-bits for vector quantization over scalar quantization. This larger gain may be explained by two factors:

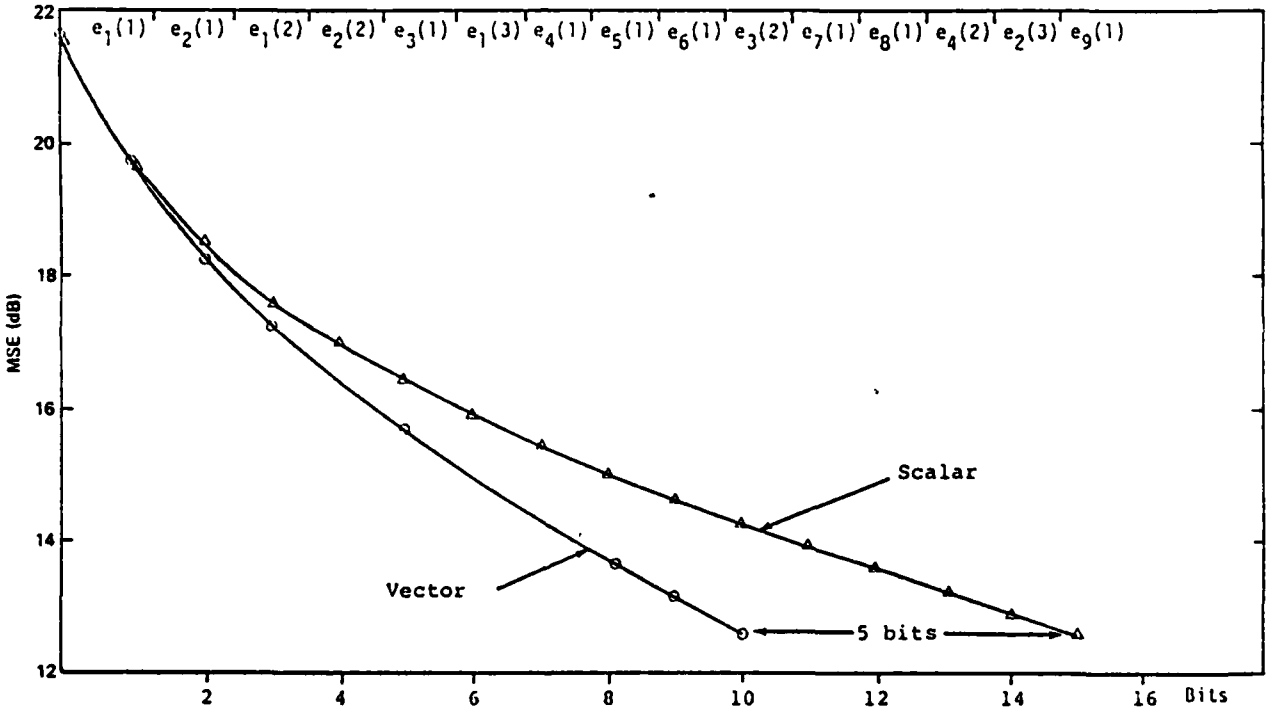


Fig. 5. Comparison of the mean-square error of vector quantization and scalar quantization. $E_i(j)$ is the ith eigenvector with an allocation of j bits.

1. The scalar quantizer used for the comparison was the minimum deviation quantizer [7]. This quantizer is suboptimal for the Itakura-Saito distance used for vector quantization. This distance measure is not separable into components so that a scalar quantizer can be designed to get the minimum distortion.

2. The parameters used for scalar quantization were not decorrelated. We have determined that the eigenvector rotation of the LARs saved 3 bits.

Even though the gain due to vector quantization is less than originally published, the reduction of 30% in the bit rate is important for the VLR vocoder and the additional complexity can be justified for this vocoder.

3.7 Cascaded Clustering

The above clustering algorithms (K-means and binary clustering) require an amount of training data that grows exponentially with the bit rate. For example, one hour of speech data is sufficient for no more than 11 to 12 bits of clustering. The above algorithms can be described as a one-stage algorithms: an input vector is quantized in one step.

To reduce the amount of training data required (in fact, we also reduce the computational load), we perform the clustering in two stages. Initially, a clustering (using either K-means or binary clustering) is performed using r bits. We refer to this

first stage as an r-bit stage. Then, each vector in the training data is quantized to the nearest template and the Quantization error vector is computed. The quantization error vector is called a deviation vector. The data set of all deviation vectors is used to perform a second stage of clustering of t bits (t-bit stage). The two sets of templates are used as a vector quantizer in the following cascaded manner. First, the nearest template to an input vector form the r-bit stage is determined. Then, the deviation (or quantization error vector) is quantized using the templates from the second t-bit stage.

The bit rate of cascaded clustering is r+t bits, yet only 2^r+2^t templates have to be estimated instead of 2^{t+r}. Therefore, both the amount of training data and the number of distance calculations in quantization are significantly reduced (both are proportional to 2^r+2^t instead of 2^{r+t}. By requiring a smaller training set and less computation than the above clustering algorithms (K-means and hierarchical clustering), the cascaded clustering method has a larger quantization error for the same bit rate. This suboptimal performance can be predicted by the following model.In cascaded clustering, we group all deviations from all the clusters together. Therefore we are implicitly assuming that all clusters of the first stage have the same deviations, i.e., all clusters have the same statistics or shape.

In other words, we are assuming that we can model the statistics of each cluster by the average statistics over all clusters. Since this is generally not true, cascaded clustering is suboptimal. Basically, by combining the deviations we are reducing the statistical dependence gain. To partially improve the performance of cascaded clustering, we increased the similarity of the clusters by using a principal component decomposition of the deviations before combining them. We represented the deviations of each cluster along the principal components of the corresponding cluster. Then we grouped all deviations. This corresponds to rotating the clusters so that their principal components align before superimposing them.

We compared several cascaded clustering algorithms on speech data, represented by 14 LAR vectors, using the Euclidean distance. Fig. 3.3 shows the mean square error of the different algorithms versus the bit rate. The 1-bit stage curve corresponds to the performance of cascaded clustering using several stages where each stage corresponds to 1-bit clustering. After 5 stages (or 5 bits) the error decreases at a rate of 6 dB/average bit, which would be obtained with optimal scalar quantization. Therefore, the statistical dependence is reduced to correlation by merging deviations for five stages. The performance of 1-bit stage cascaded clustering can be improved by

using an eigenvector rotation on the cluster as explained above. The gain due to the rotation is 2 bits, i.e., the asymptotic behavior similar to scalar quantization is delayed to seven stages. Using a 4-bit stage instead of a 1-bit stage with rotation improves performance. However, at the third 4-bit stage (or at 8 bits of cascade clustering) the slope reaches the 6 dB limit of scalar quantization. Hence, one should use the largest bit allocation to the first stage. We also have found that if we use 10 bits for the first stage, the performance of the second stage is equivalent to optimal scalar quantization. In that case, an optimal scalar quantizer may be used for the second stage instead of clustering, which indicates that no statistical dependence other than correlation is exhibited by the deviations. As we reported in Section 3, a cascaded clustering vector quantizer (10 bit vector quantizer for the first stage with a 20 bit scalar quantizer for the second stage) has the same performance as our optimal 30 bit optimal scalar quantizer. Therefore, at higher bit rates an optimal scalar quantizer would be preferred to cascaded clustering due to its simpler implementation.

The binary clustering vector quantizer has been the most effective single frame quantization method for vocoding speech from 300 b/s to 800 b/s. Typically, 10 bits per transmission

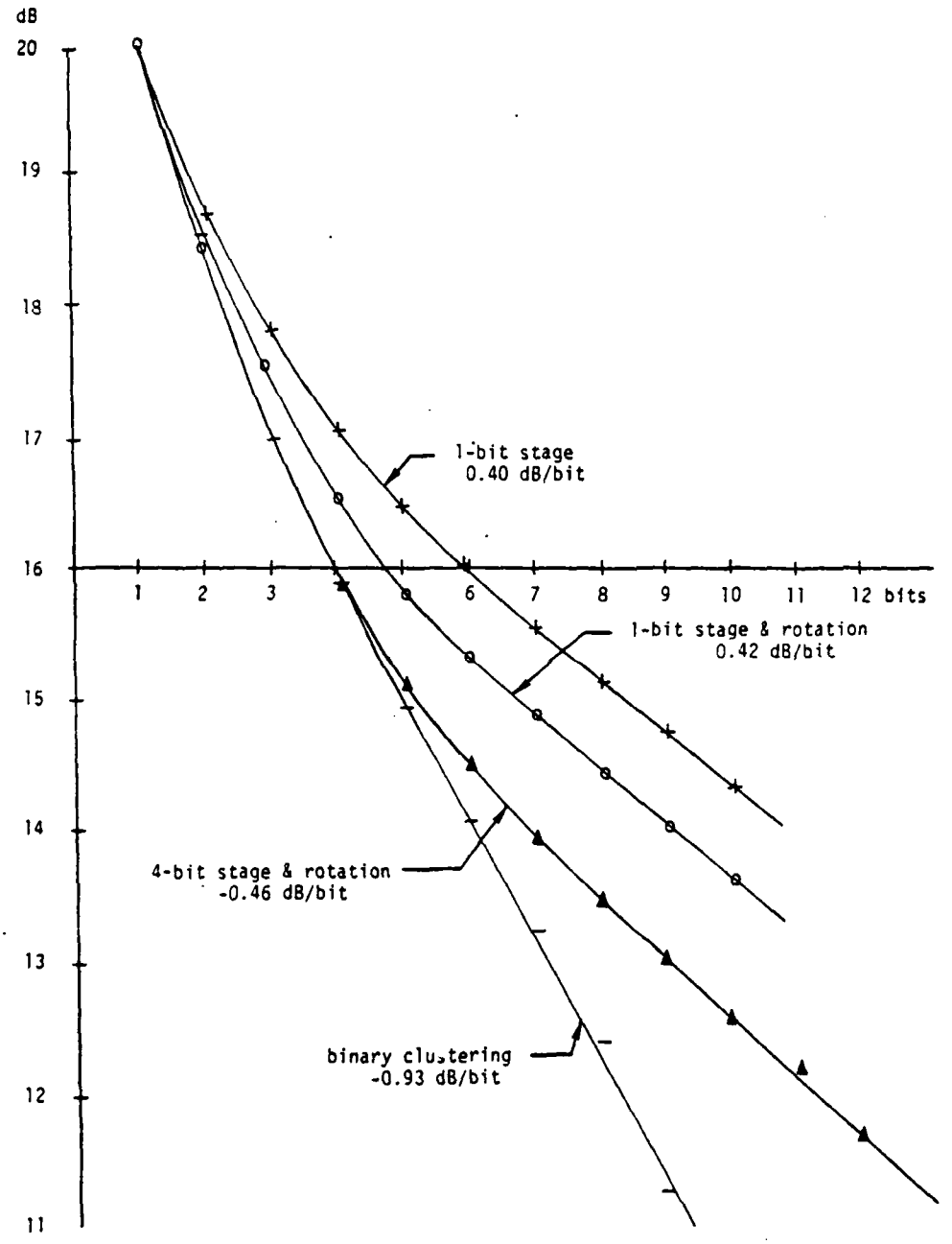


Fig. 6. Mean square error of cascaded clustering.

have been used for the spectrum. By varying the number of transmissions per second and the bit rate of pitch, gain, and voicing, we can vary the vocoder bit rate. AT 400 b/s the quality of the vocoded original is very close to 2400 b/s for a single speaker system.

We also implemented a VLR vocoder that uses a 6 bit codebook for the LPC spectrum and a VFR algorithm with an average frame rate of 30 b/s. Pitch and gain were not quantized. The output speech of this single speaker vocoder was quite intelligible. This vocoder was used for the Markov chain models of speech described in the following Section.

4. MARKOV CHAIN MODELS OF SPEECH

4.1 Introduction

We have described in the previous chapter several methods for quantizing and transmitting the spectral parameters of a single frame of speech. We have found that a vector quantizer uses the statistical dependence of the LAR parameters of a singly frame to minimize the required bit rate for vocoding speech. In particular, intelligible speech can be vocoded for a single male talker by using a 6-bit spectral codebook with a VFR algorithm that uses an average from rate of 30 b/s. The bit rate of 180 b/s for the spectral information alone was too high for the goal of a very-low-rate vocoder operating in the range of 100-200 b/s.

To reduce the bit rate of the spectral information in the above vocoder, we investigated the use of a Markov chain to model the statistical dependence of consecutively transmitted spectra, i.e., the output of the VFR algorithm was modeled as a Markov chain with an alphabet of 64 (6 bits) symbols. We evaluated 3 basic models: a first order Markov chain, a variable order Markov chain and a variable resolution Markov chain. We describe below each model. Then we present our simulation results.

If a fixed length code is used in coding the output of the

VFR algorithm, then 6 bits will be used at each transmission point. If a variable length code (entropy coding) is used, the bit rate could be reduced to the entropy of the VFR transmission sequence of 5.85 bits. We did not implement any variable length encoding, however we used entropy to compare the bit rate reductions of the several Markov models examined.

4.2 First Order Markov Chain

A Markov chain with an alphabet of M symbols (M spectral template) is characterized by a MxM transition probability matrix $[p_{ij}]$. The transition probability p_{ij} is the probability that symbol j will follow symbol i. The M^2 transition probabilities can be estimated by counting the observed transitions in a large database. For M=64 we need to estimate 4096 probabilities. Requiring an average of 10 observations for each probability, we will need 15 minutes of speech. Using a 1 hour database, the entropy of the first order chain was found to be 4.75. Hence, entropy coding and the first order model will save 1.1 bits at each transmission.

A second order chain uses the two previous symbol to predict the current symbol. We expect a 2nd order chain to have a lower entropy than a first order Markov chain. However, the amount of

data needed to estimate the 2nd order model is M times larger than that of the first order chain which would require in our case 16 hours of speech. We only have a maximum of one hour of speech. Due to the limited amount of available training data, we introduced the variable order and variable resolution Markov models [4].

4.3 Variable Order Markov Model

The set of symbols that is used to predict the following symbol is called a state. For a first order Markov chain there are M states (M=64 in our case), for a second order chain there are M^2 states (4096 states), where each state corresponds to a pair of consecutive symbols. In a variable order Markov chain we do not estimate the transition probability distribution for each state of a kth order Markov chain. Instead we determine the set of the N most probable strings of symbols of any length up to k. These are considered as the states of the Markov chain. Since these will in general have different lengths, the states of our Markov model will correspond to states of Markov chains of order from zero to k. We call this model the variable order Markov chain. The number of states N is determined by the available training data set. For a model with N=100 states, the entropy was 4.5 bits as compared with the entropy of 4.75 of a first

order chain which has 64 states. In this case, the variable order chain had 57 first order states. The states of order 2,3, and 4 were 36,5, and 2, respectively. This variable order chain had quite similar states to the order chain which explains the similarity in their entropy rate.

4.4 Variable Resolution Markov Model

Given a fixed amount of training data, we wanted to use as many high order states as possible. Since the total number of states is fixed for a given amount of training data, we used the idea of variable spectral resolution to increase the average order of the states of our Markov model. The idea of a variable resolution Markov model can best be explained by an example. A state string $x_{n-2}x_{n-1}x_n$ which is a third order state used to predict the next symbol x_{n+1}, is represented by using the following three alphabets. For the most recent symbol x_n, it uses an alphabet of size M_0 symbols, for the previous symbol x_{n-1} it uses a an alphabet of $M_1 \leq M_0$ symbols thereby using less spectral resolution to represent that spectrum. Similarly, for the oldest symbol x_{n-2} it uses M_2- symbol alphabet where $M_2 \leq M_1$ using even less spectral resolution in representing the most remote past. For a variable resolution chain of order k, there is usually an optimal combination of the size (resolution) of the alphabets M_0 thru M_{k-1} as demonstrated in [3].

We quantized 30 minutes of speech from a single male speaker using a 6 bit codebook with a VFR algorithm with an average frame rate of 30 b/s. Using this database, the optimal resolution for a 256 states variable resolution chain is given by M_0=64, M_1=32, M_2=16, M_3=8, M_4=4. The entropy of the resulting Markov chain is 3.99 bits a reduction of 1.85 bits (32%) from the zero-order model. The resulting bit rate would be 120 b/s for the spectral information alone. To achieve this bit rate variable length encoding is necessary. Since variable length encoding must be used, channel errors will have a severe effect on a vocoder based on the variable resolution model. We discuss in the next Section a more powerful method for the very-low-rate vocoding of speech that is based on segment quantization. Similarly to the Markov model, segment quantization uses the statistical dependence of consecutive spectra in speech to minimize the bit rate. But segment quantization has a more robust behavior in the presence of channed errors and does not require variable length encoding.

5. SEGMENT VOCODER

5.1 Introduction

The performance of the phonetic vocoder was only 62% correct phoneme recognition rate. We expected that a phoneme recognition rate of at least 80% is necessary for the vocoder speech to be intelligible in context. To improve the performance of the phonetic vocoder a large amount of hand-labelled speech is needed to get a better estimate of the distribution of the diphone templates. To avoid the excessively large amount of human effort to label the required large database of speech, we considered an alternate approach to the phonetic vocoder. We hypothesized that phonetic recognition may be unnecessary for the very-low-rate coding of speech in the range of 100 to 200 b/s. The diphone vocoder described in Section 2 is an example of a vocoder that does not use recognition. The segment vocoder may be considered as an extension of the diphone vocoder where speech is modeled as a sequence of segments not necessarily diphones. While a segment is analogous to a diphone it does not necessarily correspond to such a phonetic unit. Also, an automatic segmentation algorithm can be used to segment speech and avoid the extensive human effort of hand labelling required for the diphone vocoder.

An alternative viewpoint that leads to the segment vocoder is based on a model that combines the vector quantization process and the Markov model of speech. A segment which consists of a variable number of consecutive spectral frames can be quantized as a single unit. The work on Markov modeling of speech determined that consecutive spectra are highly dependent therefore not all sequences of spectra are possible. In this case is a vector quantizer that benefits from both the statistical dependence of the LARs of a single frame as well as from the statistical dependence of consecutive frames would be effective. In the segment vocoder, we exploit the statistical dependence in speech by quantizing a segment as a single unit.

5.2 Description of Segment Vocoder

In figure 5.1 we show the block diagram of the segment vocoder. The input is the unquantized LPC parameters at 100 b/s. The input is segmented with an average segment rate of 11 segments/s. Then each segment is quantized to the nearest segment template in the code book using the proper distance measure. At the receiver, the received segment templates are concatenated in sequence. A smoothing algorithm is used to reduce the spectral parameter discontinuity between adjacent segments. The resulting parameter tracks are used to drive the

usual LPC synthesizer. We describe below all the above stages of the segment vocoder in more detail and discuss the performance of the techniques used for each stage.

There are four major benefits for the segment vocoder.

1. Phonetic recognition is not necessary. The input speech is matched spectrally as closely as possible leaving the difficult task of recognizing the phoneme sequence at the receiver output to the listener.

2. Only naturally occurring sequences of spectra are used to determine the segment templates. Therefore, the segment vocoder uses the statistical dependence of consecutive spectral frames to minimize the bit rate.

3. As will be demonstrated later, the gain track and voicing pattern of a segment are highly dependent on the spectral sequence. The template gain track and voicing can be used at the receiver instead of the input's gain track and voicing. Only a level adjustment of the gain track is transmitted for each segment.

4. Finally, using naturally occurring segments as templates instead of an average template as is usual in clustering results in a crisper speech quality as discussed below. 1 This appears to hold since the timing pattern of a segment is not smeared by averaging.

5.3 Segmentation

The major advantage of the segment vocoder over the diphone vocoder is that it is completely unsupervised. We use an automatic segmentation algorithm based on spectral derivatives as discussed in [5]. We considered three types of segmentations:

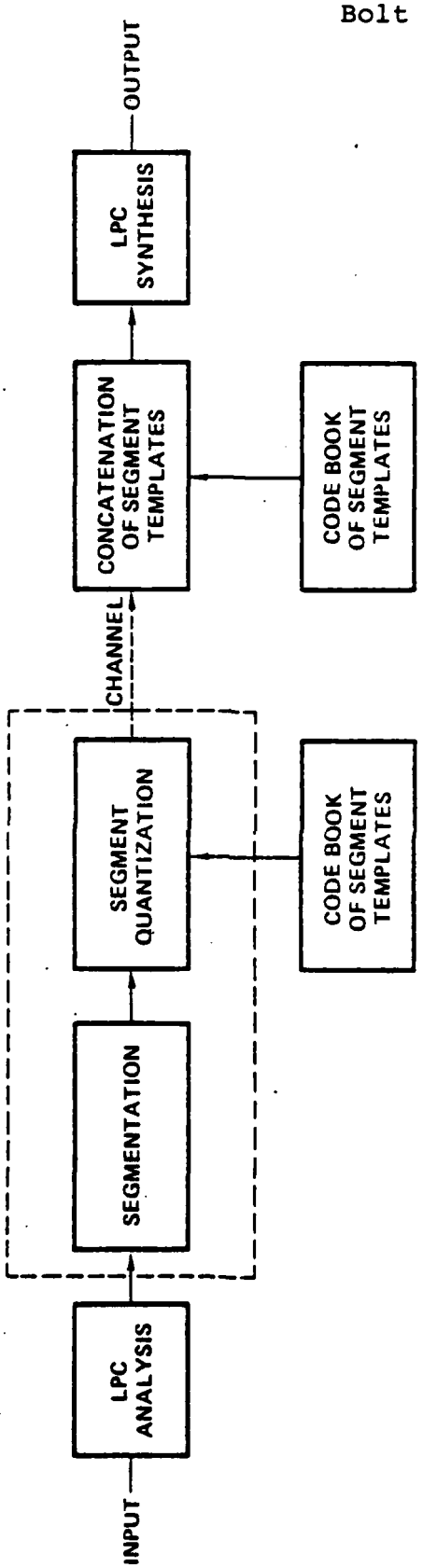


Fig. 7. Block diagram of the segment vocoder.

- Fixed length segments: each block of n frames was considered as one segment.
- Phoneme-like segments: In this case speech is considered as a sequence of steady states separated by relatively fast transitions. A phoneme-like segment was defined as the sequence of frames from the middle of a transition to the middle of the following transition.
- Diphone-like segments: A diphone-like segment is defined from the middle of a steady-state to the middle of the following steady state.

We found that the diphone-like segments have the best quality and intelligibility in our informal listening tests. We also compared the diphone-like segmentation to the true diphone segmentation as obtained by using our hand labelled database. The two segmentations resulted in the same intelligibility and quality. We therefore continued using the diphone-like segmentation for the experiments described below.

5.4 Distance Measure

Two segments will usually have different total durations. Therefore, the two segments must be time-aligned before evaluating the distance between them. Instead of using the computationally expensive dynamic time-warping approach used in isolated word recognition, we used a simple approximation called space-sampling. As described in [6], each segment is considered as a trajectory in spectral parameter space (14 LARs). The

segment is resampled at M equi-distant points along that trajectory. The corresponding spatial samples are assumed to be time aligned and the distance between two segments is the sum of the Euclidean distance between the M pairs of spatial samples. The above distance measure over-emphasizes the importance of transitions. We used a duration weighting as described in [3] to increase the importance of the steady-states portions of a segment. The contribution of each pair of space-samples was weighed by the duration of the input space-samples. In this case, the steady-states are emphasized and the vocoder speech quality improved significantly.

5.5 Input Quantization

The segment vocoder described above uses independent segmentation and quantization. The input is initially segmented with an average segment rate of 11 seg/s. Then, each input segment is quantized to the nearest template. This method of input quantization yields occasionally unintelligible segments. In this case, the segment generally encompasses several phonemes and has a large quantization error. To avoid the large quantization error of these segments we used the following method called joint segmentation and quantization.

Joint Segmentation and Quantization

In this approach, we consider all possible segmentations of the input such that the constraints on the segment durations are satisfied. We require a minimum duration of 4 frames and a maximum duration of 18 frames. For each possible input segmentation, the sequence of input segments is quantized. The segmentation that results in the smallest overall quantization error is selected as the optimal input segmentation. As described in [5], a dynamic programming search is used to implement efficiently the joint segmentation and quantization procedure. We also use a hybrid binary look-up in the segment quantization process to minimize the number of distance calculation performed.

The hybrid binary look-up was derived using the non-uniform binary clustering algorithm developed for vector quantization and described in Section 3. The binary clustering algorithm was used to divide the set of 8000 segment templates (13 bits) into 512 clusters (9 bits) each containing an average of 16 templates. Each segment template was represented using 10 space-samples and each space-sample consisted of the first 8 LARs. The limitation of 8 LARs was due to the virtual memory size limitation on our VAX computer system. The binary look-up was used to determine which cluster mean was nearest to an input spectrum. Then an

exhaustive search was used to determine the nearest template to the input from the 16 templates of the nearest cluster. This requires an average of 18+16=34 distance calculations instead of 8000, a savings of a factor of 200.

The joint segmentation and quantization method requires a large computational load of 300 times real time on the VAX when the binary look-up is used. This large computational load is justified since the resulting segment vocoder speech quality is better than the segment vocoder that uses independent segmentation and quantization. Further, the new vocoder avoids the problem of having input segments which are not well matched by a segment-template. The joint segmentation and quantization method must be used in order to satisfy the operational requirements of vocoder speech intelligibility in context.

5.6 Segment Template Selection

In the above experiments we did not specify how the set of segment templated was selected. We will readily remedy this deficiency. The set of segment templates is obtained by automatically segmenting a training database of 15 minutes of continuous speech. With average segment rate of 11 seg/s and deleting long silence intervals, we obtain 8000 segments.

All segments in this et are used as segment templates. By assuming that the set of 8000 segments is a random sample of speech segments, the above segment quantizer is a random quantizer. The performance of such a random quantizer is expected to be near-optimal in analogy to the following situation: For a Gaussian random vector, with independent components, a random quantizer can be chosen such that the expected quantization error is asymptotically equal to the distortion-rate function for a given bit rate (measured by entropy) as the dimensionality of the vector approaches infinity [7]. While the above conditions are not satisfied, we expect a random quantizer for the segment vocoder to be near-optimal because fo the large dimensionality of 140 of a segment. To determine the validity of this hypothesis, we compared the above random quantizer to a quantizer derived by using the binary clustering algorithm on segments. We compare below the two methods by segment quantization.

5.6.1 Segment Clustering

The binary clustering algorithm, used in the hybrid binary look-up described in Section 5, was used to determine a set of 8000 clusters by clustering a set of 32000 segments [8]. For each cluster of segments two types of templates were defined:

The mean segment template and the nearest to the mean segment template.

The mean segment template was obtained by averaging all the segments in a cluster using the time-alignment specified by space-sampling, i.e., the detailed timing of all segments was averaged as discussed in [8]. The nearest to the mean segment template was chosen as that segment in the cluster that was closest to the mean segment of the cluster.

We compared both types of templates to the templates of the random quantizer. The mean segment template quantizer requires 2 bits less templates than the random quantizer for the same mean square error. But for the same bit rate, we found that the random quantize has a higher quality speech than the mean segment quantizer. The higher quality speech was obtained in spite of the larger quantization error of the random quantizer. Presumably this is due to the smearing of the detailed timing in the averaging process used to obtain the mean segment template.

To avoid the smearing of the detailed timing, we used the nearest to the mean segment template. In this case we found that the random quantizer and the nearest to the mean template quantize to result in the same quantization error and the same subjective vocoded speech quality. The nearest to the mean

clustering algorithm is equivalent to the random quantizer because a very small training set is used for clustering; an average of 16 segments per cluster with a dimensionality of 140. Therefore, we will use a random quantizer for selecting a set of templates for the segment vocoder.

In the next section, we describe the methods used for quantizing the other parameters of an LPC vocoder.

5.7 Quantization of Source Parameters

An input segment is quantized to the nearest template. At the receiver, the detailed timing of the template is used for synthesis. 1 The total duration of the input segment can be quantized with 3 bit/s such that most errors are within one frame or less. The input segment duration is used to linearly scale the corresponding segment template at the receiver.

The gain track of a segment template was found to match the gain trade of the input segment, i.e., if two segments are spectrally close then their gain tracks are similar. However, a level adjustment to match the loudness of the input segment was transmitted using 2 bits. A gain normalization algorithm was used to minimize the range of the level adjustments as described in [5].

In addition to the statistical dependence of the gain track on the spectral sequence, we found that the voicing pattern of a segment to be completely specified by the spectral sequence. We do not transmit voicing in the segment vocoder. The segment template voicing is used at the receiver.

Finally, we modeled pitch by a piece wise linear model. Pitch was assumed to be linear from the middle of a transition region to the middle of the following transition region. An adaptive quantizer was used to code the change in pitch from one segment to the following segment. We used a 2 level quantizer (1 bit) with an adaptive scaling factor that is proportional to the square root of the duration between two successive transition regions.

Using the above quantization techniques, we implemented a fully coded segment vocoder that uses 20 bits for each segment and an average segment rate of 11 seg/s. We found that this vocoder can vocode speech with good quality and intelligibility with an average bit rate of 220 b/s. To reduce the bit rate further, we used a segment network analogous to the diphone network. We describe below the segment network used and the complete vocoder that operates at 150 b/s. This vocoder was demonstrated at the final ARPA NSC Meeting in June, 1982.

5.8 Segment Network

To reduce the bit rate of the segment vocoder to 150 b/s, we used a segment network to constrain the number of segment templates that can be used in quantizing the input. The segment network is analogous to the diphone network in that only a specific subset of templates is allowed to follow a segment template. For example, if the current input segment is quantized to a given template, then the following input segment must be quantized to a given template, then the following input segment must be quantized to a template that belongs to a subset of the segment templates as determined by the segment network. This subset is the set of all segment templates that follow the current template in the network.

Ideally, one should choose a network that allows all possible segment template sequences so that the quantization error is to inversed. A general method for choosing the segment network would be to determine statistically which segments are most likely to follow a given segment. This approach would require a prohibitive amount of data. We used an alternative approach based on a model that the spectral parameters of speech are continuous.

6. MULTIPLE SPEAKER SYNTHESIS

In any of the very-low-rate vocoders discussed in this report, the spectral information is reduced by removing as much redundancy as possible. One factor in reducing the information to be transmitted, is that the speech model is derived from only one speaker. For example, the speech produced by the diphone synthesis part of the phonetic vocoder sounds much like the speaker who spoke the database of diphone templates. However it is desirable for the output speech to sound like the speaker who is talking (vocoder-user). Therefore, we investigated ways of making the output of the phonetic synthesizer sound more like the speaker, without having to extract a new set of diphone templates, and using only a small amount of information that could be transmitted on the same very-low-rate transmission channel. These techniques can also be applied to the other VLR vocoders described in this report.

The basic procedure used was to require the new speaker to speak for a period of from 20 seconds to 1 minute. The material spoken could be any arbitrary text. The speech supplied was then analyzed to extract several parameters which were then used to modify the diphone templates used during synthesis, such that the speech sounded more like the vocoder-user than like the database talker.

The parameters measured and used in the transformation are the average vocal tract length (VTL) of the speaker, and the long-term average (LTA) power spectrum for voiced, unvoiced, and silence from the speaker. This procedure is described in more detail in QPR3.

6.1 Extracting Speaker Specific Parameters

The first task in this method of multiple speaker synthesis is to extract the speaker parameters from a speech sample. In experimenting with samples of varying length, we have found that at least twenty seconds of speech (excluding silences) should be analyzed in order to obtain reliable estimates for the speaker parameters.

The first parameter to extract is the average vocal tract length. This can only be reliably estimated from the formants and bandwidths during open vowels. Therefore, the program uses several heuristics to find those frames in which to measurement of VTL would yield reliable results. Specifically, it checks for voiced frames, with energy close to the local maxima, and with formant frequencies in the ranges for vowels. Furthermore, any estimates of VTL outside the range of 10-20 cm are discarded. Then the average of those accepted values is computed. A cursory

examination of the resulting averages agreed with our subjective feeling for the head size of the different talkers.

The second, and probably more important set of features extracted was the three LTA spectra for the speaker. These model the source spectrum and average vocal tract shape of the speaker. The LTA spectrum was computed separately for voiced, unvoiced, and silence spectra, since it was felt that these resulted from separate mechanisms, and therefore could vary independently.

The first task was to classify speech spectra into the three classes mentioned above. For this, we used the Acoustic-Phonetic Experiment Facility (APEF). The classifier designed was a simple linear classifier that uses as its features, the energy in the frame, relative to the 5 percentile energy, and the number of zero-crossing in the frame. Each 129-point LTA spectrum is smoothed using a 13 point raised cosine window.

We estimated that the average VTL and the three LTA spectra could be quantized and transmitted using only about 150 bits, which would take only 1.5 seconds through a 100 b/s channel.

6.2 Synthesis Using Speaker-Specific Parameters

The diphone synthesizer needs the speaker parameters of

average VTL and LTA spectra for both the database speaker, whose speech was used to create the diphone database, and for the vocoder-user speaker, whose voice the synthesizer is trying to duplicate. Given these speaker-specific parameters, and the sequence of phonemes, durations, and pitches generated by the phonetic recognizer, the phonetic synthesizer can produce speech that sounds like the vocoder user.

Each spectrum in the diphone templates used in synthesis is modified independently in the following way. Basically, each spectrum is multipled by the ratio of LTA spectra of the desired speaker, and the database speaker, for the same class of spectra. Also, the frequency axis is scaled according to the ratio of average VTL's. However, the order of these transformations is important. First, the diphone template spectrum classified as to being voiced, unvoiced, or silence. Even though we know this information from the phoneme, we use the same classifier used in the analysis of the speaker samples. The spectrum is then divided by the appropriate LTA spectrum for the database speaker to remove his speaking characteristics. Then, the frequency axis is linearly scaled according to the ratio of average VTL's of the speakers. Finally, the characteristics of the vocoder-user are inserted by multiplying by his LTA spectrum.

6.3 Evaluation of Multiple Speaker Synthesis

We have analyzed this multiple speaker synthesis process for 20 new speakers. The results of our effort in multiple speaker synthesis are encouraging. There are three main conclusions that can be drawn. First, since the phonetic vocoder transmits phoneme duration and pitch, these speaker characteristics are conveyed directly. Second, for speakers whose long-term spectra were markedly different from the database speaker, there is an audible change in the synthetic output, and the speech can sound very similar to the intended speaker. The third result is that some of the vocoder users, sound quite different from the database speaker, even though they appear to have similar LTA spectra and VTL. Therefore, the transformation does very little, and the transformed speech still sounds somewhat like the database speaker. It appears that the speaker differences for these speakers are at a more detailed level, such as the way they pronounce particular phonemes, the phase characteristics of their voice, or in the amount of nasalization they use, to name a few.

Thus, for roughly half the speakers, the transformation had the desired effect, while for the others, the speakers were similar enough that the overall changes made didn't make them more similar. In other words, the synthesizer output never

sounds very different from the vocoder-user, but it is sometimes distinguishable from speech spoken by the vocoder-user. A more detailed speaker model would necessarily require phoneme specific information to be transmitted. This could be accomplished by requiring the speaker to say a particular known passage, such that the program could extract spectra from known phonemes. These could then be used to modify the diphones associated with those phonemes.

While this method has been tested only for synthesis, it seems reasonable that the same transformation would make the recognition program more able to recognize the speech of new speakers, without extensive training to that new speaker.

REFERENCES

1. J. Makhoul, M. Krasner, S. Roukos, R. Schwartz and J. Sorensen, "Research on Narrowband Communications," Quarterly Progress Report No. 2, Contract No. F19628-80-C-0165, Bolt Beranek and Newman Inc., Tech. Report No. 4620, 18 Nov. 1980 - 17 Feb. 1981.

2. J. Max, "Quantizing for Minimum Distortion," IRE Trans. Info. Theory, Vol. IT-6, March 1960, pp. 7-12.

3. J. Makhoul, S. Roukos and R. Schwartz, "Research on Narrowband Communications," Quarterly Progress Report No. 1, Contract No. F19628-80-C-0165, Bolt Beranek and Newman Inc., Tech. Report No. 4557, 18 August - 17 November 1980.

4. J. Makhoul, S. Roucos and R. Schwartz, "Research on Narrowband Communications," Quarterly Progress Report No. 3, Contract No. F19628-80-C-0165, Bolt Beranek and Newman Inc., Tech. Report No. 4665, 18 Feb. - 17 May 1981.

5. J. Makhoul, S. Roucos and R. Schwartz, "Research on Narrowband Communications," Quarterly Progress Report No. 5, Contract No. F19628-80-C-0165, Bolt Beranek and Newman Inc., Tech. Report No. 4843, 18 Aug. - 17 Nov. 1981.

6. J. Makhoul, S. Roucos, R. Schwartz, "Research on Narrowband Communications," Quarterly Progress Report No. 4, Contract No. F19628-80-C-0165, Bolt Beranek and Newman Inc., Tech. Report No. 4766, 18 August - 17 November 1981 1981.

7. S. Roucos, R. Schwartz, and J. Makhoul, "Segment Quantization for Very-Low-Rate Speech Coding," IEEE International Conference on Acoustics, Speech and Signal Processing, Paris, France, May 1982, pp. 1565-1568

8. J. Makhoul, S. Roucos, and R. Schwartz, "Research On Narrowband Communications," Quarterly Progress Report No. 7, Bolt Beranek and Newman Inc., BBN Report No. 5089, 1 April - 30 June 1982.

Appendix

Presented at the International Conference on Acoustics, Speech, and Signal Processing, May 3-5, 1982, Paris France.

SEGMENT QUANTIZATION FOR VERY-LOW-RATE SPEECH CODING

S. Roucos, R. Schwartz, J. Makhoul

Bolt Beranek and Newman Inc.

Cambridge, MA 02238

ABSTRACT

We introduce a new method for very-low-rate vocoding that models the input speech as a sequence of variable-length segments. A segment is a sequence of frames, where each frame is represented by a spectrum, pitch and gain. We use an automatic segmentation algorithm to obtain segments with an average duration comparable to that of a phoneme. A segment is quantized as a single block. The distance measure used for quantization incorporates the appropriate time alignment of two segments. We employ a computationally efficient metric that does not use the usual dynamic programming time warping. Two basic vocoders using the above approach of block quantization have been used to transmit intelligible speech at 200 b/s.

1. INTRODUCTION

Block quantization has been used for coding the parameters of an LPC vocoder to achieve a transmission rate of 800 b/s [1]. In this paper, we describe a method based on block quantization that reduces the bit rate of an LPC vocoder to 200 b/s. Block quantization is only attractive for very-low-rate (VLR) systems for the following two reasons. First, the savings in bit rate which is usually a fixed number of bits, is most significant (percentage of bit rate) at low rates. Second, the exponential growth of the quantizer complexity with increasing bit rate makes only VLR systems practical.

The new method represents the output of LPC analysis (100 frames/s) as a sequence of segments. Each segment consists of a variable number of consecutive frames. The LPC spectra in a segment are quantized as a single block independently from other segments. We refer to this block quantizer (vocoder) as a segment quantizer (vocoder). Since we expect consecutive LPC spectra in speech to be highly dependent, only a fraction of all permutations of spectra (assumed quantized) will actually occur. Hence the bit rate of the segment vocoder will be lower than an LPC vocoder based on separately quantizing single frames.

An alternative approach that leads to the segment vocoder is based on our work on the phonetic vocoder [2]. To achieve a transmission rate around 100 b/s, we model speech by a diphone network. The nodes in the diphone network correspond to phonemes. A pair of nodes are connected by several transitions: each transition represents the LPC parameters of a typical occurrence of the diphone defined by the phoneme pair. Hence we have several templates for the same diphone. The output of the vocoder is obtained by synthesizing the best path in the network that matches the input speech. The bit rate of this vocoder is around 130 b/s but the resulting speech has not been very intelligible. To improve the intelligibility, we simplified the network. Any diphone template was allowed to follow any template (hence a sequence of templates does not necessarily correspond to a sequence of phonemes). The resulting vocoder is essentially a segment vocoder using diphones as segments. While the bit rate of this vocoder is around 200 b/s, the output speech is intelligible. Since hand labeling of speech is necessary to obtain the diphone templates, a large human effort is required to implement this vocoder. To avoid this effort, we propose to use an automatic segmentation algorithm to define the segments.

Besides the reduction in the bit rate of the spectral information over a single frame block quantizer (vector quantization), the segment vocoder achieves additional savings in coding the side information of an LPC vocoder (particularly gain and voicing). For each segment, we transmit a single pitch value, a segment duration and a gain adjustment. Since the gain track is highly dependent on the spectral sequence in a segment as shown in Section 4, gain is not transmitted. Rather, the gain track of the segment template is used and only an adjustment to the overall loudness of the segment is transmitted. Similarly, voicing is not transmitted and is obtained from the template. Another possible advantage of the segment vocoder is that the segment templates used are actual speech trajectories that have occurred. Hence, the output speech of the vocoder will have a better quality (increased naturalness) than other methods (e.g., linear interpolation in a variable frame rate vocoder). In Section 2 we present the segmentation algorithm and the distance measure used for quantization. In Section 3 we describe the vocoder, and in Section 4 we present our experimental results.

2. SEGMENTATION AND DISTANCE MEASURE

We can describe the segments of any automatic segmentation algorithm by the following three characteristics:

- o total segment duration.
- o trajectory in spectral parameter space. Each segment is viewed as a directed trajectory in parameter space where detailed timing is ignored but the direction of time is preserved.
- o detailed timing. Even if two segments have the same trajectory and total duration, they may differ in the detailed timing.

We have used the above decomposition of the segment variations in determing the coding methods of the segment vocoder, as described below. While fixed length segmentation is usually used for block quantization, the diphone model of speech suggests that a variable length segmentation might have a lower rate for the same quantization error. Fixed length segmentation will require more segment templates than variable length segmentation for the block quantizer since:

1. The lack of synchrony between fixed length segmentation and segment production in speech will produce all shifts of a given segment even if all segments have the same duration.
2. "Natural" segment durations will sometimes differ from the chosen fixed segment length resulting in segments that correspond to either pieces of "natural" segments or the concatenation of pieces of different segments. These need not occur if a proper segmentation can be used.

One can use any of several segmentation algorithms to define the variable length segments. We used a simple algorithm that considers speech as a succession of steady states separated by transitions. Two spectral time-derivatives were thresholded to determine the middle of transitions. The derivatives are:

$$d_i(n) = ||\underline{x}(n) - \underline{x}(n-i)||^2 \ , \ i=1,3 \qquad (1)$$

where $\underline{x}(n)$ is a vector of 14 log area ratios (LARs) representing the nth frame. d_1 detects fast transitions while d_3 detects slower transitions. The steady states were determined at the points of minimum d_1 within a window between two transitions. The segments were defined to begin and end in the middle of consecutive steady-states. The lower the threshold on the derivatives, the higher the segment rate. However, the distributions of the spectral derivatives are essentially bimodal (low values and high values) so that a segment rate higher than 13/s is not reasonable. We decided to use 11/s (equal to expected phoneme rate). The resulting segmentation of the automatic algorithm has been found to be generally similar to the diphone segmentation.

Distance Metric

In defining the distance measure between two segments, we have to specify the time alignment of the variable length segments. The distance measure we propose defines implicitly the required time warping. The sequence of LPC spectra in a segment represents a piecewise linear trajectory in the 14 dimensional LAR space. The total length (using a Euclidean norm on LARs) of a segment is computed and is used to define an "equi-spaced" sampled representation of the segment, i.e., the segment is

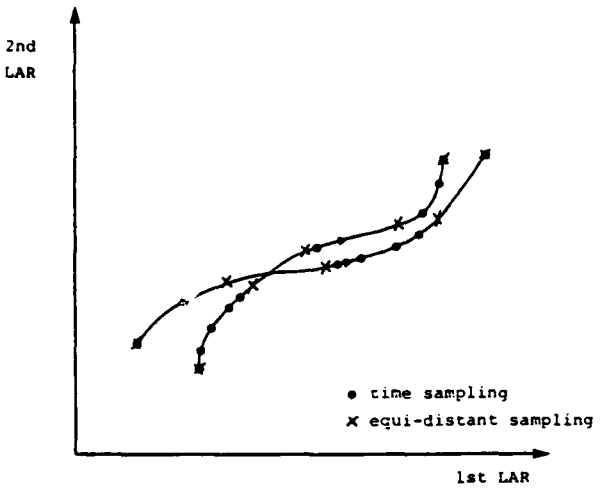


FIGURE 1. Two segments in parameter space.

resampled at a set of M equi-distant (Euclidean norm on 14 LARs) points on the trajectory. We refer to this process as spatial sampling. The distance measure, shown in Fig. 1, is similar to a metric proposed by Schroeder [3]. Given two segments with different total durations, Fig. 1, we resample both segments at M equi-distant points along their trajectories in the 14 dimensional LAR space. The distance measure between the two segments is defined as:

$$d(x,y) = \sum_{i=1}^{M} w_i ||\underline{x}_i - \underline{y}_i||^2 \qquad (2)$$

where $\underline{x}_i$, $\underline{y}_i$ are vectors of 14 LARs corresponding to the ith spatial samples of the two segments x and y, and w_i is a weight. This distance measure defines a time warping that is increasingly similar to a dynamic programming time warping as the similarity of the two segments increases. Yet, this measure is much more efficient computationally.

For each spatial (in LAR space) sample of a segment, we can associate a time of occurrence, i.e., the time when the input speech is at this point along the trajectory. We call this information the detailed timing. We define the duration of a spatial sample as the average of the two time intervals: the interval from the previous sample and the interval to the following one. We have found that a weight, w_i, proportional to the duration of a spatial sample in the distance measure improves the quantization process slightly.

To justify the above distance measure we performed the following experiment. Using the automatic segmentation algorithm at 11 segments/s, the detailed timing of each segment was modified while its total duration was preserved. The detailed timing was changed such that the time interval between consecutive spatial samples of the same segment (using M=10 samples per segment) are equal while the total duration of that segment is preserved. The resulting unquantized LPC trajectory is resynthesized. The output speech is

generally indistinguishable from the untransformed synthesized LPC. But one or two places in a 5-second sentence will have a slight problem. However, this degradation will be negligible compared to the expected degradation when the LPC parameters are quantized to 200 b/s. Hence, the detailed timing should not be used to separate two segments that have the same spectral trajectory.

3. VOCODER DESCRIPTION

We describe in this section the basic two vocoders we have evaluated, the template selection process and the quantization methods for transmitting the side information, e.g., gain, pitch, etc..

Input Segmentation

1. Separate segmentation and quantization: The sequence of LPC frames of analyzed input speech is automatically segmented at an average rate of 11 segments/s. Each segment is then quantized to the nearest segment template using the distance measure described earlier.

2. Joint segmentation and quantization: In this approach the input is not automatically segmented. Instead, all possible segmentations of the input with an average rate of 11 segments/s are considered. Then each segment is quantized using the proposed distance measure to the nearest template. The segmentation (with the corresponding quantized templates) that results in the smallest quantization error is chosen for transmission. A dynamic programming search was actually implemented to obtain the optimal joint segmentation and quantization of the input.

Template Selection

The set of segment templates is obtained by automatically segmenting (11 seg/s) a large training database of continuous speech. Each segment is a 140 dimensional vector (14 LARs x 10 spatial samples). Usually, a clustering algorithm is used to obtain an optimal set of segment templates. For the large dimensionality (140) of the segment vocoder, the expected quantization error of a properly chosen random quantizer is nearly equal to the distortion rate bound. Therefore, we do not use a computationally expensive clustering algorithm to determine the optimal set of templates. Instead, we use a random quantizer obtained by a random sample of the population of segments in speech. While this result is derived for a vector with independent components, and the corresponding optimal random quantizer is a scaled random sample [4], we expect the above random quantizer for the segment templates to be as effective.

Side Information

To complete the description of the vocoder, we present the methods adopted to quantize gain, voicing, pitch and timing. Since the detailed timing is not perceptually important for the vocoded speech, the detailed timing of the template is used. The total duration of a segment is quantized with 3 bits such that a peak error of 1 frame is allowed and real time is preserved as closely as possible (the sum of all transmitted durations equals the sum of the durations of the segments of the input speech).

Voicing information is not transmitted. The sequence of voicing decisions is determined from the segment template. Pitch is transmitted once per segment using an adaptive quantizer for the increment in pitch from the previous segment. The increment is obtained by the best linear fit for the pitch track. The adaptive quantizer uses 3 bits and increases the size of the nonuniform steps as an increasing function of the segment duration. The gain track of the template is used at the receiver. The gain track of the templates was normalized to compensate for changes in the loudness level. This normalization reduced the gain quantization error. However, a 2-bit adjustment to the gain track is transmitted to equalize the means (in dB) of the input segment and the nearest template. We found that a gain normalization of the gain track of the templates improved gain quantization. The normalization was done by compensating for the changes in the overall loudness level of the speech used for the templates. At the receiver, the parameters were smoothed at the junction of consecutive segments. The bit rate for all the side information was 88 b/s. In Table 1, we summarize the bit allocation used in quantizing the different parameters.

	Bit Allocation
Spectral Segment	13 bits
Gain Adjustment	2
Pitch	3
Duration	3
	21 bits/segment

Bit Rate = 21 x 11 seg/s = 231 b/s

Table 1: Bit Allocation

4. EXPERIMENTAL RESULTS

The database used in evaluating the segment vocoder consisted of 15 minutes of continuous speech from a single male speaker reading a textbook. This data was automatically segmented at an average rate of 11/s. The resulting 9000 (~13 bits) segments were used as the segment templates of the random quantizer. Another set of 5 sentences from the same speaker was vocoded to determine the intelligibility and quality of the vocoder. Each sentence was six seconds long.

The first set of experiments compared the following 3 systems:

1. Fixed Length Segmentation: Both the database for the templates and the input were segmented into fixed length segments of 9 frames (or 11 frames/s). This vocoder does not transmit duration.

2. Variable Length Segmentation: The automatic segmentation algorithm was used to segment

both the database and the input at an average rate of 11/s. The segment duration varied between 4 and 18 frames.

3. Joint Segmentation and Quantization: The database was automatically segmented at an average rate of 11/s. Dynamic programming, as described in Section 3, was used to obtain the best segmentation and quantization of the input. However, to reduce the computational load, we used a binary lookup for segment quantization instead of the exhaustive search used in the first two systems. The binary lookup was defined by performing an 8-bit binary clustering on the 13-bit templates. Each cluster had an average of 5 bits of templates. An input segment was quantized to the nearest cluster (each cluster was represented by its mean segment), then an exhaustive search of all the templates in that cluster was used to determine the nearest template to the input.

The side information (gain, pitch, duration and voicing) was coded in the same manner for all three vocoders. The bit rate was 200-230 b/s. The output speech of all three vocoders was quite intelligible. While the second vocoder had a slightly higher quality (less roughness) than the first, it occasionally (once per sentence) missed one or two phonemes. The reason for missing phonemes in the second vocoder is that several phonemes were lumped as one long segment. The third vocoder is significantly better than the first two. In fact, it does not suffer from the problem of lumping phonemes into one long segment.

To determine the degradation in performance caused by the binary lookup, we used it in the second vocoder instead of the exhaustive search. The quantization error using the binary lookup with 13 bits of templates was equal to the quantization error of an exhaustive search quantizer using 10.5 bits (a loss of 1.5 bits which is quite audible). Hence, the optimal segmentation of the third vocoder not only compensates for this loss, but results in a larger improvement since it is better than the first two vocoders.

In Fig. 2, we show the input and output parameters of the second segment vocoder. In this figure total duration is not quantized to preserve synchrony. However, the detailed timing of the template is used for the output. The voicing decisions and pitch values of the output of the segment vocoder match very well with the input. The strong dependence of voicing on the spectral segment is a major benefit of the segment vocoder approach. Gain is another example of the advantage of the segment vocoder due to the strong dependence on the spectral segment. The gain track is well predicted except for a level adjustment. We also show the first two log area ratios to illustrate the detailed match (instead of a piecewise linear match) that contributes to the smoother quality of the segment vocoder output.

5. CONCLUSION

We have shown that block quantization can be used to vocode intelligible speech at 200 b/s. Automatic segmentation based on spectral derivatives was demonstrated to be as effective as diphone segmentation (done by hand). The major advantages of the segment vocoder are good quality

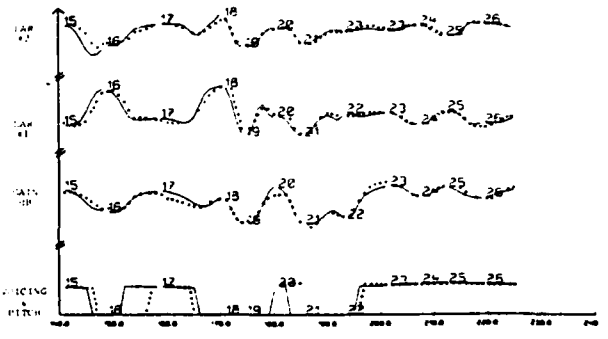


FIGURE 2. Unquantized parameters (solid) and quantized parameters (dotted) for 1 s of speech. The variable length segmentation is also shown.

output speech (due to naturally occurring templates) and the efficiency in quantizing the side information (in particular gain and voicing). We are currently investigating methods to reduce the bit rate to 150 b/s with minimal loss in intelligibility. We are constraining the segment templates to define a network analogous to the diphone network. However, instead of using phonemes to constrain the diphone templates, as in the diphone network, a threshold on the spectral distance from the end of a segment to the beginning of another determines which segments can follow it.

ACKNOWLEDGMENT

This work was supported by the Advanced Research Projects Agency and monitored by RADC under Contract No. F19628-80-C-0165.

REFERENCES

(1) D.W. Wong, B.H. Juang, and A.H. Gray, Jr., "Recent Developments in Vector Quantization for Speech Processing," International Conf. on Acoustics, Speech, and Signal Processing, pp. 1-4, Atlanta, GA, March 1981

(2) R. Schwartz, J. Klovstad, J. Makhoul, and J. Sorensen, "A Preliminary Design of a Phonetic Vocoder based on a Diphone Model," International Conf. on Acoustics, Speech, and Signal Processing, pp.32-25 , Denver, CO, April 1980.

(3) M.R. Schroeder, "Similarity Measure for Automatic Speech and Speaker Recognition," JASA, Vol. 43, No. 2, pp. 375-377, 1968.

(4) D.J. Sakrison, "A Geometric Treatment of the Source Encoding of a Gaussain Random Variable," IEEE Trans. Inform Theory, IT-14, pp. 481-486, 1968.

Presented at the International Conference on Acoustics, Speech, and Signal Processing, May 3-5, 1982, Paris, France.

A VARIABLE-ORDER MARKOV CHAIN FOR CODING OF SPEECH SPECTRA

S. Roucos, J. Makhoul, R. Schwartz

Bolt Beranek and Newman Inc.

Cambridge, MA 02238

ABSTRACT

We present a method that reduces the bit rate of a low rate LPC vocoder by modelling the sequence of quantized spectra by a Markov chain. To minimize the bit rate, one would want to use a high-order chain. Unfortunately, a high-order chain would require an inordinate amount of data for training. We describe in this paper the use of a variable order Markov chain that maximizes the effective use of a given amount of speech data.

To reduce the number of states of a high-order chain, we define an equivalence relation on the states, i.e., "similar" states are grouped together in an equivalence class and a single conditional distribution is associated with the equivalence class. We introduce two equivalence relations. In the first, called variable order Markov chain, the equivalence classes represent the most probable states of any order. In the second method, called variable resolution, the equivalence class is obtained by decreasing the quantization accuracy in representing a spectrum that belongs to a more remote past. For an LPC vocoder with 64 possible spectra (using 6-bit vector quantization), the second method is superior to the first and decreases the entropy from 6 bits to 4 bits per spectrum with 256 equivalence classes.

1. INTRODUCTION

We have recently developed a variable frame rate (VFR) LPC vocoder that uses block quantization (vector quantization) [1] for quantizing the log area ratios (LARs) of an LPC spectrum. This vocoder is a single speaker system that transmits the LARs at a rate of 180 b/s: The average frame rate is 30 f/s and the LARs are quantized using 6 bits. In order to obtain a vocoder that operates below 200 b/s, we must reduce the bit rate of the spectral information to allow us to transmit gain, pitch, voicing and timing information. We modelled the output sequence of the VFR algorithm using several models to achieve more efficient encoding of the output. The simplest model was a first-order Markov chain which reduced the bit rate to 128 b/s for the spectral information (from 6 bits to 4.25 bits per transmission). Higher order models were expected to further reduce the bit rate. However, since a limited amount of "training" data was available, we had to restrict the type of models that can be employed. We present below two general classes of models that may be used for efficient encoding of the VFR output sequence. We then present the models we chose and describe some experimental results.

2. SOURCE MODELS

There are two general classes of models that have traditionally been used for modelling discrete information sources. Recently, Rissanen [2] demonstrated that one class, the recursive models, is in fact superior to the other, the alphabet extension models. He showed that for the same complexity (measured by the number of probabilities that specify the model), a recursive model can always be found that has at least the same entropy as an alphabet extension model. He also showed that the converse is not true. We describe below both models.

Recursive Models

Let s be a string (a finite length sequence) of symbols from an alphabet S. We assume that the alphabet S has N symbols (in our case, N spectral templates). The probability of a string $s=x_1x_2\ldots x_n$ of length n is given by:

$$P(s) = P(x_1)\, P(x_2|x_1)\ldots P(x_n|x_1x_2\ldots x_{n-1}). \quad (1)$$

The class of recursive models is defined by constraining the conditional probabilities P(x|s) in the following manner. Let S^* denote the set of all finite strings over the alphabet S. Suppose that the set S^* is partitioned into K equivalence classes that are defined by a function f:

$$f: S^* \rightarrow Z \quad \text{where } Z = \{1,2,\ldots K\} \quad (2)$$

The conditional probabilities of a recursive model are required to satisfy

$$P(x|s) = P(x|f(s)). \quad (3)$$

In other words, the conditional probability of the symbol x depends only on the equivalence class of the string s. This model is specified by K(N-1) conditional probabilities. The optimal average code length required for encoding the information source that corresponds to the above model is given

by the entropy h:

$$h = -\sum_{i=1}^{K} P_i \sum_{j=1}^{N} P(j|i) \log P(j|i) \quad (4)$$

where P_i is the probability of the ith equivalence class and j is the jth symbol of the alphabet and the base of the logarithm is two.

Usually the conditional probabilities that specify the recursive model are estimated using an observed output sequence of the information source. The maximum likelihood estimate of the conditional probability is

$$\hat{P}(j|i) = \frac{n(j|i)}{n(i)} \quad (5)$$

where n(j|i) is the number of times symbol j occurs just after the equivalence class i has occurred, and n(i) is the number of times the ith equivalence class occurs in the observed sequence. For a long sequence, the random variable $n_i(P(j|i)-\hat{P}(j|i))$ is asymptotically Gaussian with zero mean and with variance $P(j|i)(1-P(j|i))$ [3]. Hence, the estimate is asymptotically unbiased and consistent.

Alphabet Extension

Another class of models that can be used to model the output sequence is based on alphabet extension. In this approach, one defines new symbols to represent a group (string) of symbols. Usually the new symbols of the extended alphabet are used to represent highly probably strings. The underlying assumption is that the model with the extended alphabet "captures" the behavior of the source and hence should decrease the bit rate necessary to encode it.

There are two factors that may reduce the bit rate when alphabet extension is used. First, the addition of new symbols changes the probability structure. Second, the average duration between successive symbols increases since the added symbols represent concatenations of the original symbols. To evaluate the reduction in bit rate, we compare the bit rate of a zero memory model of an information source using either the original alphabet S_N (N letters represented by the integers 1,2,...N) or an extended alphabet S_{N+1} (N+1 letters). We do not assume that the source is zero memory but that the model used for coding it is. Assume that the new symbol N+1 represents the string 12 (1 followed by 2). Let p_1, p_2 be the probabilities of occurrence of 1 and 2 respectively and let p_{12} be the joint probability of 12 in that order. The bit rate r_{N+1} using S_{N+1} is

$$r_{N+1}=(1-p_{12})(r_N+F(p_{12},1)-F(p_1,p_{12})-F(p_2,p_{12})) \quad (6)$$

where $F(x,y)=(x-y)\log(x-y)-x\log x$, and r_N is the rate of the model using S_N. It is relatively easy to show that r_{N+1} could be larger than r_N. Hence, increasing the model complexity (adding one more symbol to the alphabet) may increase the bit rate. This cannot happen with the recursive model. One can see that the problem with zero memory alphabet extension is that higher order statistics must be estimated using a recursive model, which Rissannen [2] showed can then be substituted by a recursive model without alphabet extension. We do not pursue the class of alphabet extension models any further.

3. MARKOV CHAIN MODELS

First-Order Markov Chain

To reduce the general recursive model to a first-order Markov chain, we require each equivalence class to correspond to one symbol of the alphabet S of the chain. Therefore, we have N equivalence classes, where N is the alphabet size. For a first-order Markov chain, the equivalence class of a string s is the class represented by the rightmost symbol of s. The transition probabilities of this model satisfy:

$$P(x_{n+1}|s) = P(x_{n+1}|x_n) \quad (7)$$

where $s=x_1,x_2\ldots x_n$. Therefore, the probability distribution of the next symbol x_{n+1} depends only on the current symbol x_n, called the state of the chain. To specify this model, we need to estimate N(N-1) conditional probabilities.

Variable Order Markov Chain

The entropy of a Markov chain is monotone nonincreasing with the order of the chain. But, the order of the chain one can estimate is severely limited by the amount of training data required. For a kth order chain, $(N-1)N^k$ transition probabilities must be estimated. For a second order chain, with k=2, and N=64 symbols, we need 20 hours of speech (at 30 f/s) to estimate the transition probabilities (requiring only 10 observations/transition). Since the number of states grows exponentially with the order of the chain, we use equivalence classes on the states to reduce the number of conditional probabilities one must estimate.

The equivalence classes are defined such that each represents a unique state of variable order. A state of order k is the string of the k most recent symbols, i.e., at time n the kth order state is the string $x_{n-k+1}x_{n-k+2}\cdots x_n$. We will use the words equivalence class and state interchangeably. The collection of states (or equivalence classes) that we are considering can be grouped into a state tree (Fig. 1). Each node of the tree corresponds to a state. Each node, except the root node, has a label which is a symbol from the alphabet of the information source. The state defined by a node is the string of symbols obtained by traversing the tree from that node to the root node. The root node corresponds to the equivalence class of all other states not accounted for by the other nodes of the tree.

time	0	1	2	3	4	5
VFR, x_n	a	b	a	c	d	a
state, s_n	a	null	aba	c	null	ad
tree node	2	1	6	3	1	5

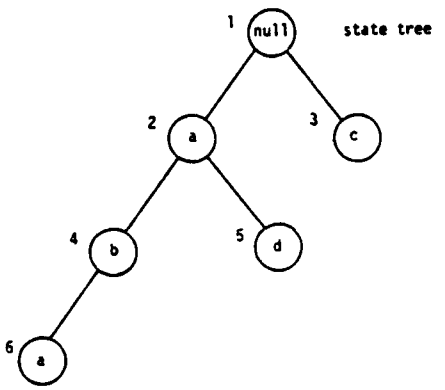


FIG. 1. State tree: We show a sequence of symbols, their state sequence and the corresponding nodes on the tree.

We present in the next section a method for generating a state tree and estimating the corresponding constrained Markov model. However, we should stress that the state tree representation does not allow all possible state sets. For every string that is a state, the state tree requires that all its prefixes are also states of the model.

Variable Order Model Estimation - The approach is to sequentially add states to the state tree until the required number of states has been reached. The algorithm consists of the following:

1. Initialize the state tree to have one node only: the null state.

2. Using the training data, estimate the conditional probability distributions of all states currently in the tree.

3. Test for highly probable state-symbol pairs. We used a count of 30 for a specific state transition pair as a threshold (the training data size was an average of 10 counts/pair). Let s_n and x_{n+1} be such a pair. Create a new state $s'=s_n x_{n+1}$ obtained by concatenating x_{n+1} and s_n.

4. When the number of created states equals the required number of states, stop adding states and reestimate the conditional probabilities using all the training data set. Otherwise, go to Step 2.

We implemented the above algorithm with two variations. In Step 2, it is not clear how much training data should be used before going to Step 3. To see the difficulty, we note that the transition counts for recently created states will be underestimated as compared to older states. One method is to loop through steps 2, 3, and 4 for every observed symbol. Another method is to analyze a block of data, then create a set of new states, then zero all estimates and go to step 2 again. The latter method, though computationally more expensive, results in a model with slightly lower entropy (by 0.1 bit).

Variable Resolution States - A state of the variable order Markov model is an equivalence class used to condition the next symbol. The purpose of the modelling is to find the minimal number of equivalence classes (or states) needed to condition speech to get the lowest entropy. One method of decreasing the number of equivalence classes with minimal loss in the effectiveness of state conditioning is based on a variable spectral resolution representation of the classes (states). The idea is that strings that differ only in the "remote" past by small distances should belong to the same equivalence class. We are assuming that a distance between the symbols is available. In the case of the VFR output sequence, a spectral distance is used. One method to implement the above is to use a different codelength (set of spectral templates) for the symbols in a state string that depends on the position of the symbol in the string. The codelength decreases as the position corresponds to a more distant past. For example, let $s=x_{n-2}x_{n-1}x_n$ be a state string. Then x_n may have 64 possible values (6 bits), x_{n-1} 32 values (5 bits) and x_{n-2} 16 values (4 bits). On the state tree, this means that the number of possible labels of a node depends on the level of the node in the state tree. This number decreases as the level of the node increases.

4. EXPERIMENTAL RESULTS

Initially, we estimated a first-order Markov chain for a fixed rate (100 f/s) LPC vocoder that uses a 6-bit vector quantizer. The entropy decreased from 5.50 for a zero memory source model, to 2.25 for a first-order model. We also estimated a variable order model with 64 states which reduced the entropy to 2.13, a small improvement over a first-order chain. However, the bit rate for the fixed frame rate system is still high (213 b/s for the spectral information only). To reduce the bit rate, we used a piecewise linear model for the trajectory of the LPC vectors (linear in LARs) to determine the transmission points of a VFR system [4]. In addition to the reduction in bit rate, the VFR vocoder speech is much smoother than that of the fixed-rate vocoder. A single speaker system using an average frame rate of 30 f/s and a 6-bit vector quantizer yields quite intelligible speech. Using the above vocoder, we analyzed 30 minutes of continuous speech from a single male talker. The output sequence of the VFR algorithm was used to estimate several recursive models. The variable resolution models used a 6-bit hierarchical (binary tree) vector quantizer to define several quantizers with decreasing resolution (6,5,4,...bits). Table 1 shows the different models estimated and their

		1st Order Chain	Variable Order and Resolution					
			R=65432	R=65432	R=65432	66432	54321	43211
Entropy		4.25	4.39	4.15	3.99	4.88	4.54	4.64
Number of Conditioning Classes		64	64	128	256	32	32	32
Distribution of State Order	1	64	47	62	64	29	25	15
	2		14	53	132	2	6	11
	3		2	10	37			4
	4			2	14			1
	5				6			
Model Complexity		4032	2907	4752	7431	1651	1624	1493

Table 1. Parameters of Markov Models.

corresponding entropies. The zero memory entropy of the VFR vocoder was 5.85 bits. For the variable order and resolution models, we define the variable R that specifies the spectral resolution used at each level of the state tree (position in time along the state). A value of R=65432 means that a 6-bit quantizer is used at level 1 (time n), a 5-bit quantizer at level 2 (time n-1), and so on. For these models, we also show the distribution of the number of states used with a given order (from 1 to 5). For the variable order models, the root node corresponds to a state of order zero (zero memory state). The complexity of the models considered, as determined by the number of transition probabilities estimated, is shown in Table 1. For the first-order Markov chain, the worst case complexity is shown. For the other models, the actual number of conditional probabilities estimated is indicated. In the case of the VFR output sequence, the variable order model with 64 states is slightly worse than a first-order model. The reason for this difference is that highly probably states are not as effective as the set of all first-order states. To improve the performance of the variable order model, we tried another method for selecting the states on the state tree which are be extended. The states with the largest contribution to the average code length were extended first. The performance of these models was similar to those that used the most probable states for extension.

To illustrate the performance of the variable resolution model, we considered a model with 32 states. We chose different spectral resolutions for defining the states as shown in columns 5 through 7 of Table 1. For this low number of states (32), we found that decreasing the resolution to an optimal value (R=54321) has the lowest entropy of 4.54. We also found that as the number of states is increased, the required optimal resolution increases. Finally, the entropy is monotone decreasing with the number of states, as shown in columns 2 through 4. For a model with 256 states, the entropy is reduced from 5.85 to 3.99, a savings of 1.86 bits.

5. CONCLUSION

In this paper we described the use of recursive models to reduce the bit rate of the spectral information to 120 b/s (using 256 states) in a VFR vocoder. The flexibility of the models (continuous increase of the number of states) allows a more efficient use of the available data than the usual fixed order Markov chains. However, the selection of the equivalence classes is arbitrary. One cannot guarantee that the resulting classes are optimal (minimal entropy). Further work is needed to determine a criterion for selecting which states must be extended.

ACKNOWLEDGMENTS

This work was supported by the Advanced Research Projects Agency and monitored by RADC under Contract No. F19628-80-C-0165. The authors wish to thank Dr. M. Berouti for many stimulating discussions.

REFERENCES

(1) A. Buzo, A.H. Gray, Jr., R.M. Gray, et al, "Speech Coding Based on Vector Quantization," IEEE Trans., Vol. ASSP-28, pp. 562-574, Oct. 1980.

(2) J. Rissanen, and G.G. Langdon, Jr., "Universal Modelling and Coding," IEEE Trans. Inform. Theory, IT-27, No. 1, pp. 12-22, 1981.

(3) U.N. Bhat, "Elements of Applied Stochastic Processes," John Wiley & Sons, New York, 1972.

(4) E. Blackman, R. Viswanathan, W. Russell and J. Makhoul, "Narrowband LPC Speech Transmission over Noisy Channels," Proc. ICASSP-79, pp. 60-63, April 1979.

VECTOR QUANTIZATION FOR VERY-LOW-RATE CODING OF SPEECH

S. ROUCOS, R. SCHWARTZ AND J. MAKHOUL

Bolt Beranek and Newman Inc.
Cambridge, MA 02238

ABSTRACT

We describe in this paper several vector quantization techniques that can be used to transmit speech at a bit rate ranging from 150 b/s to 800 b/s. The methods can be grouped into two classes: single frame quantization methods and segment quantization methods. In single-frame quantization methods, all the parameters of the vector being quantized represent a single frame of speech (typically 20 msec). In segment quantization, the parameters of the vector being quantized represent speech events on the order of a phone (90 ms). We present three frame quantization methods based on clustering algorithms and compare their performance to optimal scalar quantization. Then we describe the new segment quantization method that can be used to transmit intelligible speech at 150 b/s.

1. INTRODUCTION

For narrowband speech compression, the LPC vocoder achieves reasonable quality and intelligibility at a bit rate of 2400 b/s. In the LPC vocoder, we quantize the log-area-ratio (LAR) parameters using a scalar quantization method. In scalar quantization each LAR is quantized separately. To reduce the bit rate of the LPC vocoder, Buzo [1] proposed to use vector quantization for quantizing the LPC spectrum. In this method, all the linear prediction coefficients that represent an input speech spectrum are considered as a vector and quantized as a single unit. Using vector quantization one can reduce the bit rate to 800 b/s with a small decrease in the quality of the vocoded signal.

We describe in this paper several methods for vector quantization and evaluate their performance for the very-low-rate transmission of speech. We consider the range from 100 b/s to 800 b/s. The vector quantizers that we describe fall in two classes:

1. Single frame Quantization: In these methods, the parameter vector that is quantized represents the spectrum of a single frame of speech. These vector quantization methods have been used from 300 b/s to 800 b/s.

2. Segment Quantization: In this case, the parameter vector represents a sequence of speech spectra. Typically, a segment of the input speech with a duration comparable to that of a phoneme is quantized as a single unit. We have used this novel method of segment quantization for vocoding speech from 150 b/s to 250 b/s (single speaker vocoder).

In Section 2, we define vector quantization. In Section 3, we describe several vector quantizers for single frame quantization and compare their performance to optimal scalar quantization. In Section 4, we introduce a new approach for very-low-rate vocoding of speech based on segment quantization.

2. VECTOR QUANTIZATION

In this section we describe two necessary conditions for optimal vector quantization. In vector quantization we combine several parameters into a single vector and quantize all parameters simultaneously, instead of quantizing each separately. The n-dimensional vector $\underline{x}$ is used to represent a set of n parameters. An M-level, n-dimensional block quantizer is defined by a partition $P=\{C_i;\ i=1,M\}$ of the space of all input vectors into M regions, each denoted by C_i. A template vector z_i is also defined for each region C_i. The input vector $\underline{x}$ is quantized by determining the region C_i that contains $\underline{x}$, and the template z_i of that region is used as the quantized value of $\underline{x}$. This block quantizer has been called a vector quantizer, a term which will be used in this paper to indicate that a single-frame quantization method is used, i.e., the vector represents a single frame (typically 20 ms) of speech.

A nonnegative distortion measure, denoted by $d(\underline{x},\underline{z})$, is used as an objective measure of the loss in accuracy in representing an input vector $\underline{x}$ by a template $\underline{z}$. An optimal vector quantizer quantizes an input vector using the minimum distance classification rule:

$$\underline{x} \in C_i \iff d(\underline{x},z_i) \le d(\underline{x},z_j), \quad 1 \le j \le M. \qquad (1)$$

The templates of the optimal vector quantizer must be the center of mass of their corresponding region C_i, i.e., the template z_i minimizes

$$\int_{C_i} d(\underline{x},\underline{z})p(\underline{x})d\underline{x}, \qquad (2)$$

where $p(\underline{x})$ is the probability density function of $\underline{x}$ assumed to exist. The above two optimality conditions were initially used by Lloyd to design optimal one-dimensional (scalar) quantizers. In pattern recognition, Macqueen [2] derived the K-means clustering algorithm using the above two optimality conditions. Buzo [1] used the K-means clustering algorithm for an LPC vocoder operating at 800 b/s.

In the following section we present several algorithms for deriving a vector quantizer using clustering techniques.

3. SINGLE-FRAME QUANTIZATION

In this section, we describe three methods for vector quantization that have been used for coding the LPC spectrum of an input speech frame. All three methods use a clustering algorithm on a training set of input vectors for designing the vector quantizer. The clustering algorithms differ in the amount of training data needed, the computational and memory requirements, and the resulting quantization error. The three methods are

- o K-means clustering
- o Binary clustering
- o Cascaded clustering

We evaluate the performance of the above algorithms and compare it with optimal scalar quantization. We then discuss their usefulness for very-low-rate coding of speech.

K-Means Algorithm

The K-means algorithm has been used extensively in pattern recognition. Using a training set of observed vectors representing speech spectra, the K-means algorithm is a hill-climbing algorithm that determines a set of K templates that minimize the clustering criterion. The clustering criterion is the average quantization error. In our case, we need to find K = M templates. It is based on the optimality conditions described in Section 2. Below, k is the iteration index and $z_i(k)$ is the estimate of the template of cluster C_i at iteration k. The steps of the algorithm are:

1. Initialization: Set k=0. Choose by some adequate method a set of M initial spectral templates $z_i(1)$ for $1 \le i \le M$.
2. Classification: k<--k+1. Classify all the training data $\underline{x}$ to the corresponding nearest template. This defines the clusters $C_i(k)$:

$$\underline{x} \in C_i(k) \text{ iff } d(\underline{x}, z_i(k)) \le d(\underline{x}, z_j(k)) \qquad (3)$$

$1 \le j \le M$

3. Template Updating: Update the template of every cluster using all model spectra assigned to that cluster in Step 2. For cluster i, the new template $z_i(k+1)$ is the vector $\underline{z}$ that minimizes the cluster average distance given by

$$D(C_i(k)) = \frac{1}{M_i(k)} \sum_{\underline{x} \in C_i(k)} d(\underline{x}, \underline{z}) \qquad (4)$$

4. Termination Test: If the templates $z_i(k+1)$ are significantly different from $z_i(k)$, go to Step 2; otherwise, stop.

Duda and Hart [3] present several methods for obtaining an initial set of templates. The above algorithm can be shown to converge. However, the K-means algorithm may converge to a local optimum. A classical solution to get global optimality has been to use different sets of initial templates (Step 1), and then to choose the best final result.

There are two reasons to justify the use of a vector quantizer instead of a simple scalar quantizer:

1. Using the same distortion measure, an optimal vector quantizer has a smaller distortion than an optimal scalar quantizer for the same bit rate. This advantage increases as the statistical dependence of the parameters increases. However, as we show later, for the mean-square error distortion measure, only the statistical dependence that is different from correlation will contribute to a difference between vector and scalar quantization.
2. The second advantage of vector quantization is the ability to use vector distortion measures that cannot be minimized by a scalar quantizer such as the Itakura distortion measure. However, the usefulness of such a vector measure must be justified. In particular, we have found no difference between the Itakura distance measure and the simple Euclidean distance on LARs when compared using 10-bit vector quantizers. The vocoded speech using the two quantizers were informally compared and resulted in similar quality. Therefore, we will use the simple Euclidean distance on LARs in the remainder of this paper.

To determine the gain of vector quantization over scalar quantization due to statistical dependence, we performed several experiments that we describe below. We initially define optimal scalar quantization.

Optimal Scalar Quantization

We first describe the optimal scalar quantization process for a Gaussian random vector of parameters. Then we compare the mean-square error of the K-means algorithm and the optimal scalar quantizer in quantizing LPC parameters of speech.

Given a set of n parameters represented by a random vector $\underline{x}$, and a fixed number of bits b, the optimal scalar quantizer that minimizes the mean-square error consists of three steps:

1. Parameter decorrelation
2. Bit allocation
3. Scalar quantization.

We describe each of these steps below.

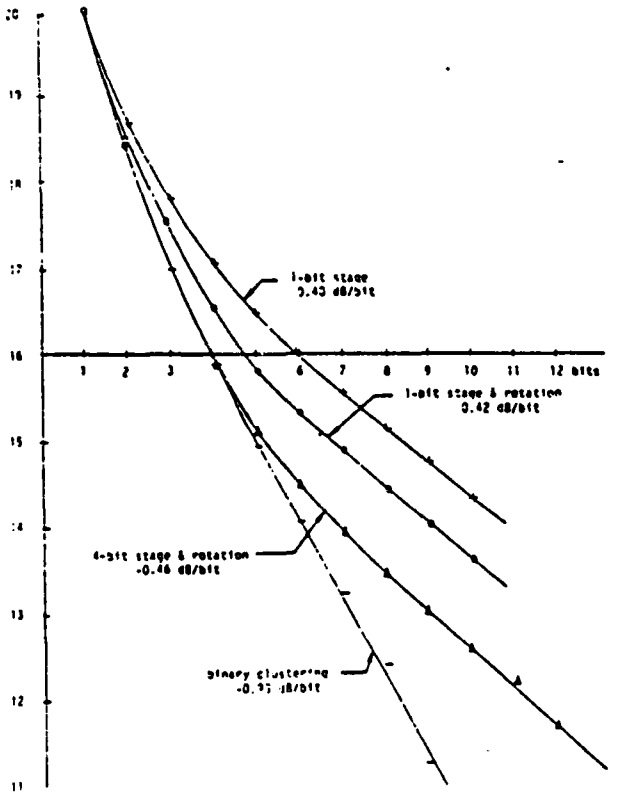
Parameter decorrelation: Let Q be the matrix whose columns are the eigenvectors of the covariance matrix C of the Gaussian random vector $\underline{x}$. The new parameter vector $\underline{y} = Q'\underline{x}$ will have uncorrelated components.

Bit allocation: The second step is to allocate the given b bits among the components of $\underline{y}$. Segall [4] derived the optimal bit allocation for the Euclidean distance measure.

Scalar Quantization: The third and final step is to perform the scalar quantization of each component y_i using b_i bits as allocated in the previous step. Here one simply uses a Max quantizer [5] designed for each component.

In the application of optimal scalar quantization to LAR quantization, one estimates the covariance C and the corresponding transformation matrix Q from a set of training speech samples.

We compared the performance of optimal scalar quantization and vector quantization for randomly generated vectors with a Gaussian distribution. We used a training sequence of 15,000 vectors to compare the performance of both quantizers from 1 to 10 bits with the dimensionality varying from 10 to 14. The covariance matrix was chosen to be the same as that of the LAR vectors of speech spectra. The performance of both quantizers as measured by the mean-square error and the entropy of the



By grouping the deviations from all the clusters together, we are implicitly assuming that all clusters of the first stage have the same deviations (same statistics or shape), and that we can model the statistics of each cluster by the average over all clusters. Since this is generally not true, cascaded clustering is suboptimal. Basically, by combining the deviations we are reducing the statistical dependence gain. To partially improve the performance of cascaded clustering, we increased the similarity of the clusters by using a principal component decomposition of the deviations before combining them. We represented the deviations of each cluster along the principal components of the corresponding cluster. Then we grouped all deviations. This corresponds to rotating the clusters so that their principal components align before superimposing them.

We compared several cascaded clustering algorithms on speech data, represented by 14 LAR vectors, using the Euclidean distance. Fig. 2 shows the mean square error of the different algorithms versus the bit rate. The 1-bit stage curve corresponds to the performance of cascaded clustering using several stages where each stage corresponds to 1-bit clustering. After 5 stages (or 5 bits) the error decreases at a rate of 6 dB/average bit, which would be obtained with optimal scalar quantization. Therefore, the statistical dependence is reduced by merging deviations. The performance of 1-bit stage cascaded clustering can be improved by using an eigenvector rotation on the cluster as explained above. The gain due to the rotation is 2 bits. Using a 4-bit stage instead of 1 bit with rotation improves performance. However, at the third 4-bit stage (or at 8 bits of cascaded clustering) the slope reaches the 6 dB limit of scalar quantization. Hence, one should use the largest bit allocation to the first stage. We also have found that if we use 10 bits for the first stage, the performance of the second stage is equivalent to optimal scalar quantization. In that case, an optimal scalar quantizer may be used for the second stage instead of clustering. As we reported in Section 3, a cascaded clustering vector quantizer (10 bit vector quantizer for the first stage with a 20 bit scalar quantizer for the second stage) has the same performance as our optimal 30 bit scalar. Therefore, at higher bit rates an optimal scalar quantizer would be preferred to cascaded clustering.

The binary clustering vector quantizer has been the most effective single frame quantization method for vocoding speech from 300 b/s to 800 b/s. Typically, 10 bits per transmission have been used for the spectrum. By varying the number of transmissions per second and the bit rate of pitch, gain, and voicing, we can vary the vocoder bit rate. At 400 b/s the quality of the vocoded original is very close to 2400 b/s in a single speaker system. In the next section, we describe a new approach called segment quantization that can be used for transmitting speech at 150 b/s.

4. SEGMENT QUANTIZATION

We presented in Section 3 several vector quantizers based on clustering techniques. In this section, we discuss a new vector quantizer that does not use clustering. Instead, the set of templates is obtained by a random sampling technique. Before discussing the performance of such a vector quantizer, we describe the structure of the segment vocoder [8] that uses random quantization for transmitting speech using a bit rate from 150 b/s to 250 b/s.

The gain of vector quantization over scalar quantization increases as the amount of the statistical dependence of a set of parameters increases. Since we expect consecutive speech spectra to be highly dependent, a vector quantizer that quantizes a sequence of spectra as a unit would be most effective. In segment quantization we use a segment which consists of a variable number of consecutive frames as the unit for quantization. Below, we present the segmentation algorithm used to define the segments, the distance measure between two segments, the segment template selection process and a brief description of the segment vocoder.

Segmentation

Our work on the phonetic vocoder [9] provides a basis for selecting what events in speech must be combined into one segment. In the phonetic vocoder, we model speech by a diphone network. A diphone is represented by a sequence of LPC spectra from the middle of a phoneme to the middle of the following phoneme. However, hand labeling of speech is necessary to obtain the diphone templates, requiring a large human effort. To avoid this effort, we propose to use an automatic segmentation algorithm to define the segments.

One can use any of several segmentation algorithms to define the variable length segments. We used a simple algorithm that considers speech as a succession of steady states separated by transitions. Two spectral time-derivatives were thresholded to determine the middle of transitions. The derivatives are:

$$d_i(n) = ||\underline{x}(n) - \underline{x}(n-i)||^2 \ , \ i=1,3 \qquad (5)$$

where $\underline{x}(n)$ is a vector of 14 LARs representing the nth frame. d_1 detects fast transitions while d_3 detects slower transitions. The steady states were determined at the points of minimum d_1 within a window between two transitions. The segments were defined to begin and end in the middle of consecutive steady-states. We decided to use an average segment rate of 11/s (equal to expected phoneme rate). The resulting segmentation of the automatic algorithm has been found to be generally similar to the correct diphone segmentation.

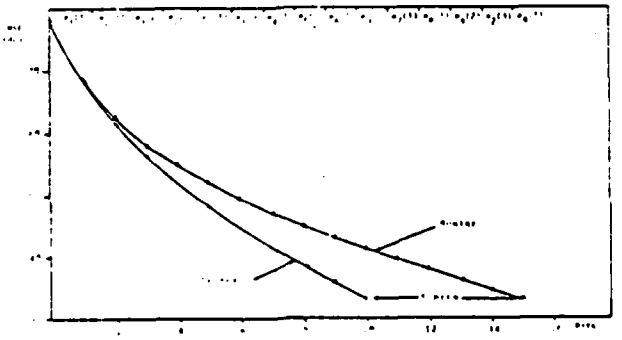
Distance Measure

In defining the distance measure between two segments, we have to specify the time alignment of

quantizer output was almost identical (both within 3%). Therefore, for a low dimensionality (~15) one should simply use the optimal scalar quantization process for a Gaussian random vector.

Statistical Dependence

The major justification for using vector quantization instead of scalar quantization for speech compression has been based on the expected superior performance due to the statistical dependence of speech spectral parameters. We have seen that parameter correlation does not contribute to a difference in performance between vector and optimal scalar quantization. Hence, we have to determine if speech exhibits any statistical dependence other than correlation to justify the use of vector quantization. To estimate the savings in bit rate due to statistical dependence, we compared a vector quantizer with an optimal scalar quantizer for a data base of speech spectra represented by 14 LARs. The Euclidean distance was used to measure the quantization error. Fig. 1 shows the mean-square error of both quantizers. We also show in Fig. 1 the bit allocation used for the optimal scalar quantization. For each additional bit, we show the eigenvector that gets this additional bit and the cumulative sum of bits allocated to that component. We found that the vector quantizer was better than the scalar quantizer. The mean-square error of the 10-bit vector quantizer was equal to that of the 15-bit optimal scalar, a saving of 5 bits.



The advantage of vector quantization over optimal scalar quantization (a gain of 5 bits for the same mean-square error) is most significant for very-low-rate vocoding of speech. Practical limitations on the amount of computing and training data limits optimal vector quantizers to about 10 to 12 bits. For higher bit rates, suboptimal vector quantizers such as cascaded clustering, which is described below may be used. However, the resulting loss in optimality reduces the advantage of vector quantization over optimal scalar. When we compared the two methods (cascaded and scalar) at 30 bits, we found vector quantization to be less robust than optimal scalar which resulted in the same performance for both methods. Therefore, at these higher bit rates, a scalar quantization method would be most effective.

Recent published results [6] using the Itakura-Saito distance claim an advantage of 14 bits for vector quantization! This larger gain may be explained by two factors:

1. The scalar quantizer used for the comparison was the minimum deviation quantizer [7]. This quantizer is suboptimal for the Itakura-Saito distance used for vector quantization. This distance measure is not separable into components so that a scalar quantizer can be designed to get the minimum distortion.

2. The parameters used for scalar quantization were not decorrelated. We have determined that the eigenvector rotation of the LARs saved 3 bits.

Binary Clustering

The K-means clustering algorithm has an extensive computational load for both the training (clustering phase) and the quantization phase. The extensive load is due to the exhaustive search that requires M (for M templates) distance calculations to determine the nearest template. A binary clustering procedure reduces the number of distance calculations to $2\log_2 M$ by imposing hierarchical structure on the clusters. This procedure is equivalent to defining a tree structure whose nodes correspond to clusters.

The binary clustering is applied sequentially in the following manner on a training data set. Initially, the binary clustering algorithm divides the training data set of model spectra into two clusters using the K-means algorithm with K=2. Then each cluster is further subdivided into two clusters until the desired number of clusters is obtained. The K-means algorithm (where K=2) is always used in dividing a given cluster.

There are two issues of the binary clustering technique that we consider in more detail: (1) how to choose which clusters to subdivide next, and (2) how the quantization error of a binary quantizer compares with that of an optimal quantizer.

There are several methods for selecting which cluster to subdivide next. The uniform tree method divides all clusters at a given level in the tree. Therefore, the resulting binary tree is uniform. Another method is to select that cluster that has the largest contribution to the quantization error. This method results in a nonuniform tree.

Using the mean-square error (on LARs), we compared the uniform binary clustering, nonuniform binary clustering, and the K-means clustering. The nonuniform binary clustering required 0.5 bits less than the uniform binary clustering for the same quantization error. Further, the nonuniform binary clustering required 0.5 bits more than the K-means algorithm for the same MSE.

The minimal loss in performance of the nonuniform binary clustering can be tolerated in most applications given the tremendous savings in computation (only $2\log_2 M$ distances on the average instead of M).

Cascaded Clustering

The above clustering algorithms (K-means and binary clustering) require an amount of training data that grows exponentially with the bit rate. For example, one hour of speech data is sufficient for no more than 11 to 12 bits of clustering. The above algorithm can be described as a one-stage algorithm: an input vector is quantized in one step.

To reduce the amount of training data required (in fact, we also reduce the computational load), we can use cascaded clustering. The idea is to perform the clustering in two stages. Initially, a clustering (using either K-means or binary clustering) is performed using r bits. We refer to this stage as an r-bit stage. Then, the deviation from the nearest template (quantization error vector) for all the data in the training set are computed. The data set of deviations is used to perform a second stage of clustering of t bits (t-bit stage). The two sets of templates are used as a vector quantizer in the following cascaded manner. First, the nearest template to an input vector from the r-bit stage is determined. Then, the deviation (or quantization error vector) is quantized using the templates from the second t-bit stage.

The bit rate of cascaded clustering is r+t bits, yet only 2^r+2^t bits templates have to be estimated instead of 2^{t+r}. Therefore, both the amount of training data and the number of distance calculations in quantization are significantly reduced (both are proportional to 2^r+2^t instead of 2^r2^t).

the variable length segments. The distance measure we propose defines implicitly the required time warping.

The sequence of LPC spectra in a segment represents a piecewise linear trajectory in the 14 dimensional LAR space. The total length (using a Euclidean norm on LARs) of a segment is computed and is used to define an "equi-spaced" sampled representation of the segment, i.e., the segment is resampled at a set of M equi-distant (using the Euclidean norm on 14 LARs) points on the trajectory. We refer to this process as spatial sampling. The distance measure is similar to a metric proposed by Schroeder [10]. Given two segments with different total durations we resample both segments at M equi-distant points along their trajectories in the 14 dimensional LAR space. Then, the distance measure between the two segments is defined as:

$$d(x,y) = \sum_{i=1}^{M} w_i ||x_i - y_i||^2 \qquad (6)$$

where x_i, y_i are vectors of 14 LARs corresponding to the ith spatial samples of the two segments x and y, and w_i is a weight. This distance measure is more efficient than a distance measure that uses a dynamic programming time warping.

Template Selection: Random Quantization

The set of segment templates of the segment vocoder is obtained by automatically segmenting 11 seg/s) a large training database of continuous speech. We consider this set of templates a: a randomly selected set and call it a random quantizer. We do not use a clustering algorithm to determine the set of templates for the segment quantizer because of the excessive amount of training data required and computatio.al load. However, we expect the performance of the two methods to be similar because of the large dimensionality of the vector representing a segment (140 dimensions). In fact, for a Gaussian vector with independent identically distributed components, one can show that a set of N templates obtained by a random sample of N vectors, has an expected mean-square error equal to the optimal distortion rate function [11] and therefore is optimal.

Since segments do not have independent Gaussian components, we determined the loss in optimality when a random quantizer is used instead of a segment quantizer based on binary clustering of segments. We found that the cluster based quantizer requires 2 bits less than the random quantizer for the same mean-square error, a small savings compared to the complexity of segment clustering. Further, we found that for the same bit rate, the random quantizer results in a better subjective speech quality than clustering despite a larger quantization error. We believe that the averaging process used to determine a template in clustering smears the detailed trajectories of the segments which results in a more muffled speech.

Vocoder Description

We describe in this section the basic segment vocoders. The sequence of LPC frames of analyzed input speech is automatically segmented at an average rate of 11 segments/s. Each segment is then quantized to the nearest segment template using the distance measure described earlier.

To complete the description of the segment vocoder, we present the methods adopted to quantize gain, voicing, pitch and timing. These methods are described in detail in [8]. The total duration of a segment is quantized and transmitted. Voicing information is not transmitted. The sequence of voicing decisions is determined from the segment template. Pitch is transmitted once per segment using an adaptive quantizer that uses the best linear fit of the pitch track. The gain track of the template is used at the receiver. However, a 2-bit level adjustment to the gain track is transmitted to equalize the means (in dB) of the input segment and the nearest template. At the receiver, the parameters were smoothed at the junction of consecutive segments. The segment vocoder can transmit intelligible speech at 220 b/s for a single speaker. Using a segment network we can reduce the bit rate to 150 b/s with a minimal loss in quality [12].

5. CONCLUSION

Vector quantization techniques are useful for vocoding speech at bit rates varying from 150 b/s to 800 b/s. For higher bit rates, a simple optimal scalar quantizer is preferred. For the lower rates from 150 b/s to 250 b/s, a segment vector quantizer based on random quantization was demonstrated to be effective for the transmission of speech.

1. A. Buzo, A.H. Gray Jr., R.M. Gray, and J.D. Markel, "Speech Coding Based Upon Vector Quantization," IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-28, Oct. 1980, pp. 562-574.

2. J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Data," Proceedings of the 5th Berkeley Symposium on Probability and Statistics, University of California Press, Berkeley, 1967.

3. R.O. Duda and P.E. Hart, "Pattern Classification and Scene Analysis," John Wiley, New York, 1973.

4. A. Segall, "Bit Allocation and Encoding for Vector Sources," IEEE Trans. Inform. Theory, Vol. IT-22, March 1976, pp. 162-169.

5. J. Max, "Quantizing for Minimum Distortion," IRE Trans. Info. Theory, Vol. IT-6, March 1960, pp. 7-12.

6. D.Y. Wong, B.H. Juang and A.H. Gray, Jr., "Recent Developments in Vector Quantization for Speech Processing," Int. Conf. on Acoustics, Speech, and Signal Processing, Atlanta, GA, March-April 1981, pp. 1-4.

7. J.D. Markel and A.H. Gray, Jr., "Implementation and Comparison of Two Transformed Reflection Coefficient Scalar Quantization Methods," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-28, No. 5, Oct. 1980, pp. 575-583.

8. S. Roucos, R. Schwartz, and J. Makhoul, "Segment Quantization for Very-Low-Rate Speech Coding," IEEE International Conference on Acoustics, Speech and Signal Processing, Paris, France, May 1982, pp. .

9. R. Schwartz, J. Klovstad, J. Makhoul, and J. Sorensen, "A Preliminary Design of a Phonetic Vocoder Based on a Diphone Model," IEEE International Conference on Acoustics, Speech and Signal Processing, Denver, CO, April 1980, pp. 32-35.

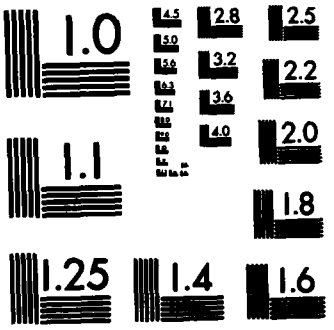
10. M.R. Schroeder, "Similarity Measure for Automatic Speech and Speaker Recognition," Journal of the Acoustical Society of America, No. 2 1968, pp. 376-377.

11. D.J. Sakrison, "A Geometric Treatment of the Source Encoding of a Gaussain Random Variable," IEEE Trans. Inform. Theory, Vol. IT-14, No. 3, May 1968, pp. 96-101.

12. J. Makhoul, S. Roucos, and R. Schwartz, "Research On Narrowband Communications," Quarterly Progress Report No. 7, Bolt Beranek and Newman Inc., BBN Report No. 5089, 1 April - 30 June 1982.

AD-A122 838 RESEARCH ON NARROWBAND COMMUNICATIONS(U) BOLT BERANEK 2/2
AND NEWMAN INC CAMBRIDGE MA S ROUCOS ET AL. NOV 82
BBN-5231 F19628-80-C-0165

UNCLASSIFIED F/G 17/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

FISCAL STATUS REPORT

Contract F19628-80-C-0165-BBN Job No. 08280

Status of Funds Report

FINAL REPORT

In accordance with the provisions of Section F of the Contract Schedule, the following is submitted as the Status of Funds Report.

Reporting Date, as of	November 30, 1982
Contract Value	$765,806
Amount Funded to	765,806
Amount Committed	763,920
Uncommitted Balance	1,886
Estimated Cost to Complete	1,886

As of the above date, the Contractor does not anticipate any cost overrun in the performance of the subject contract.

RESEARCH ON NARROWBAND COMMUNICATIONS

Research and Development Status Report
Final Report

ARPA order No. 3515, AMD. 4

Contract No. F19628-8Ø-C-Ø165

Name of Contractor:
Bolt Beranek and Newman Inc.

Principal Investigator:
Dr. John Makhoul
(617) 497-3332

Effective Date of Contract:
18 August 198Ø

Contract Expiration Date:
3Ø November 1982

Sponsored by

Defense Advanced Research Projects Agency (DoD)
Monitored by RADC/EEV

END

FILMED

2-83